

AR035873-1-MA-nur

Lecture Notes in Computer Science

1386

Thomas A. Henzinger
Shankar Sastry (Eds.)

Hybrid Systems: Computation and Control

First International Workshop, HSCC'98
Berkeley, California, USA, April 1998
Proceedings

19990316 067

DTIC QUALITY INSPECTED 1



Springer

August 31, 1999

Memo to ~~XXXXXXXXXX~~

Per my voice message telephone conversation this morning, I'm requesting the following changes be made on a final report submitted to DTIC receiving the following AD number:

ADA361 329

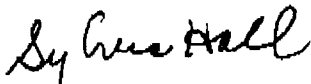
Changes to be made on the Standard Form 298

Change Block 5 to read DAAG55-98-1-0259

Change Block 10 to read ARO 38745.1-MA-CF

If you have any questions to this request please call me at DSN 832-4220.

Thank you



Sylvia Hall
U.S. Army Research Office
RTP, NC 27709-2211

REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comment regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE 2/17/99	3. REPORT TYPE AND DATES COVERED Final Report	
4. TITLE AND SUBTITLE Hybrid Systems: Computation and Control			5. FUNDING NUMBERS DAAH04-96-1-0341	
6. AUTHOR(S) Thomas A. Henzinger and Shankar Sastry (Eds.)				
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(ES) Regents of the University of California c/o Sponsored Projects Office 336 Sproul Hall Berkeley, CA 94720-5940			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211			10. SPONSORING / MONITORING AGENCY REPORT NUMBER ARO 35873.91-MA-MUR	
11. SUPPLEMENTARY NOTES The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.			12 b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) This volume contains the proceedings of the <u>First International Workshop on Hybrid Systems: Computation and Control</u> , HSCC'98, organized April 13-15, 1998, at the University of California, Berkeley. The focus of the workshop is on mathematical methods for the rigorous and systematic design and analysis of hybrid systems. A hybrid system consists of digital devices that interact with analog environments. Driven by rapid advances in digital controller technology, hybrid systems are objects of investigation of increasing relevance and importance. The emerging area of hybrid systems research lies at the crossroads of computer science and control theory: computer science contributes expertise on the digital aspects of a hybrid system, and control theory contributes expertise on the analog aspects. Since both research communities speak largely different languages, and employ largely different methods, a major purpose of the workshop is to bring together researchers from both disciplines. The workshop will also include demonstrations of software tools for the design, analysis, and simulation of hybrid systems.				
14. SUBJECT TERMS hybrid systems-hierarchical, nonlinear, and safety-critical; computer science; control theory; digital devices; analog environments; timed regular languages.			15. NUMBER OF PAGES 415	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OR REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

Springer

Berlin

Heidelberg

New York

Barcelona

Budapest

Hong Kong

London

Milan

Paris

Santa Clara

Singapore

Tokyo

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

Thomas A. Henzinger
Shankar Sastry
University of California at Berkeley
Department of Electrical Engineering and Computer Sciences
Berkeley, CA 94720, USA
E-mail: {tah,sastry}@eecs.berkeley.edu

Cataloging-in-Publication data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Hybrid systems : computation and control ; first international workshop ; proceedings / HSCC '98, Berkeley, California, USA, April 13- 15, 1998. Thomas A. Henzinger ; Shankar Sastry (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Budapest ; Hong Kong ; London ; Milan ; Paris ; Santa Clara ; Singapore ; Tokyo : Springer, 1998

(Lecture notes in computer science ; Vol. 1386)
ISBN 3-540-64358-3

CR Subject Classification (1991): C.1.m, C.3, D.2.1,F.3.1, F.1.2, J.2

ISSN 0302-9743

ISBN 3-540-64358-3 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

© Springer-Verlag Berlin Heidelberg 1998
Printed in Germany

Typesetting: Camera-ready by author
SPIN 10632061 06/3142 - 5 4 3 2 1 0 Printed on acid-free paper

Preface

This volume contains the proceedings of the *First International Workshop on Hybrid Systems: Computation and Control*, HSCC'98, organized April 13–15, 1998, at the University of California, Berkeley. Following several meetings that were initiated by Anil Nerode at Cornell University, this is the first of a newly constituted, regular annual series of workshops on hybrid systems. Papers from the earlier meetings were published in the Springer-Verlag Lecture Notes in Computer Science series, volumes 736, 999, 1066, 1201, and 1273. The steering committee of the new workshop series includes Panos Antsaklis (University of Notre Dame), Nancy Lynch (Massachusetts Institute of Technology), Amir Pnueli (Weizmann Institute, Israel), Alberto Sangiovanni-Vincentelli (University of California, Berkeley), and Jan van Schuppen (CWI, The Netherlands).

The focus of the workshop is on mathematical methods for the rigorous and systematic design and analysis of hybrid systems. A hybrid system consists of digital devices that interact with analog environments. Driven by rapid advances in digital controller technology, hybrid systems are objects of investigation of increasing relevance and importance. The emerging area of hybrid systems research lies at the crossroads of computer science and control theory: computer science contributes expertise on the digital aspects of a hybrid system, and control theory contributes expertise on the analog aspects. Since both research communities speak largely different languages, and employ largely different methods, a major purpose of the workshop is to bring together researchers from both disciplines.

The three-day workshop will feature six invited keynote speakers and 26 contributed talks that were selected from 55 submissions by a technical program committee. The keynote lecturers will be Panos Antsaklis (University of Notre Dame), Stephen Boyd (Stanford University), Edward Lee (University of California, Berkeley), Alberto Sangiovanni-Vincentelli (University of California, Berkeley), Joseph Sifakis (VERIMAG, France), and Murray Wonham (University of Toronto). Additional invited addresses will be given by Linda Bushnell from the Army Research Office and by Helen Gill from the Defense Advanced Research Projects Agency. The workshop will also include demos of software tools for the design, analysis, and simulation of hybrid systems.

The program committee was chaired by the editors and included Rajeev Alur (University of Pennsylvania), Karl Astrom (Lund University, Sweden), Albert Benveniste (INRIA-IRISA, France), Ahmed Bouajjani (VERIMAG, France), Michael Branicky (Case Western Reserve University), Peter Caines (McGill University), Datta Godbole (PATH Berkeley, California), Mark Greenstreet (University of British Columbia), Vineet Gupta (NASA Ames, California), Bruce Krogh (Carnegie Mellon University), Stephane Lafortune (University of Michigan), Kim Larsen (Aalborg University, Denmark), Oded Maler (VERIMAG, France), Stephen Morse (Yale University), Anil Nerode (Cornell University), Peter Ramadge (Princeton University), Roberto Segala (University of Bologna,

Italy), and Howard Wong-Toi (Cadence Berkeley Labs, California). In the selection process, the program committee was aided by the following reviewers: L. Aceto, K. Al-Wahedi, E. Asarin, E. Badouel, G. Barrett, O. Bournez, A. Chutinan, P. Codognet, R. Debouk, A. Deshpande, A. Fehnker, A. Hicks, R. Jagadeesan, M. Kourjanski, Y. Lakhnech, F. Lin, J. Lygeros, H. McClamroch, R. Nikoukhah, G. Pappas, A. Puri, R. Rajamani, H. Schumacher, R. Sengupta, A. Skou, M. Sorine, C. Tomlin, and C. Weise. The steering committee handled all submissions that were co-authored by the program chairs.

We are grateful to all invitees, contributors, and reviewers for making the workshop a success. In addition, we wish to thank Carol Block for administrating the workshop organization, John Lygeros and Serdar Tasiran for organizing the tool demos, Alexa Brudy and Flora Oviedo for organizational support, and the Army Research Office for generous financial support.

January 1998

Thomas A. Henzinger
Shankar Sastry

Table of Contents

E. Asarin <i>Equations on Timed Languages</i>	1
A. Balluchi, M. Di Benedetto, C. Pinello, C. Rossi, A. Sangiovanni-Vincentelli <i>Hybrid Control for Automotive Engine Management: The Cut-Off Case</i>	13
A. Beydoun, L.Y. Wang, J. Sun, S. Sivashankar <i>Hybrid Control of Automotive Powertrain Systems: A Case Study</i>	33
S. Bornot, J. Sifakis <i>On the Composition of Hybrid Systems</i>	49
L. Bushnell, O. Beldiman, G. Walsh <i>An Equivalence between a Control Network and a Switched Hybrid System</i>	64
B. Carlson, V. Gupta <i>Hybrid cc with Interval Constraints</i>	80
T. Dang, O. Maler <i>Reachability Analysis via Face Lifting</i>	96
A. Fehnker <i>Automotive Control Revisited: Linear Inequalities as Approximation of Reachable Sets</i>	110
E.D. Ferreira, B.H. Krogh <i>Switching Controllers Based on Neural-Network Estimates of Stability Regions and Controller Performance</i>	126
V. Friesen <i>A Logic for the Specification of Continuous Systems</i>	143
M.R. Greenstreet, I. Mitchell <i>Integrating Projections</i>	159
K.X. He, M.D. Lemmon <i>Lyapunov Stability of Continuous-Valued Systems under the Supervision of Discrete-Event Transition Systems</i>	175
T.A. Henzinger, V. Rusu <i>Reachability Verification for Hybrid Automata</i>	190
G. Lafferriere, G.J. Pappas, S. Sastry <i>Subanalytic Stratifications and Bisimulations</i>	205
G. Lehrenfeld, R. Naumann, R. Rasche, C. Rust, J. Tackén <i>Integrated Design and Simulation of Hybrid Systems</i>	221
E.S. Lemch, P.E. Caines <i>Hierarchical Hybrid Systems: Partition Deformations and Applications to the Acrobot System</i>	237

C. Livadas, N.A. Lynch <i>Formal Verification of Safety-Critical Hybrid Systems</i>	253
J. Lygeros, N.A. Lynch <i>Strings of Vehicles: Modeling and Safety Conditions</i>	273
J. Lygeros, G.J. Pappas, S. Sastry <i>An Approach to the Verification of the Center-TRACON Automation System</i>	289
Z. Manna, H.B. Sipma <i>Deductive Verification of Hybrid Systems Using STeP</i>	305
A.S. Matveev, A.V. Savkin <i>Reduction and Decomposition of Differential Automata: Theory and Applications</i>	319
B.M. Miller <i>Optimization of Generalized Solutions on Nonlinear Hybrid (Discrete-Continuous) Systems</i>	334
T.W. Neller <i>Information-Based Optimization Approaches to Dynamical System Safety Verification</i>	346
C. Tomlin, J. Lygeros, S. Sastry <i>Synthesizing Controllers for Nonlinear Hybrid Systems</i>	360
J.H. van Schuppen <i>A Sufficient Condition for Controllability of a Class of Hybrid Systems</i>	374
X. Li, T. Zheng, J. Hou, J. Zhao, G. Zheng <i>Hybrid Regular Expressions</i>	384
M. Žefran, J.W. Burdick <i>Stabilization of Systems with Changing Dynamics</i>	400
List of Authors	417

Equations on Timed Languages *

Eugene ASARIN

Institute for Information Transmission Problems
19 Bolshoi Karetnyi lane, 101447, Moscow, Russia
asarin@ippi.ras.ru

Abstract. We continue investigation of languages, accepted by timed automata of Alur and Dill. In [ACM97] timed regular expressions equivalent to timed automata were introduced. Here we introduce quasilinear equations over timed languages with regular coefficients. We prove that the minimal solution of such an equation is regular and give an algorithm to calculate this solution. This result is used to obtain a new proof of Kleene theorem ([ACM97]) for timed automata. Equations over timed languages can be also considered as an alternative way of specifying these languages.

1 Introduction

Timed automata ([AD94]) form the best investigated class of hybrid systems. It is known which problems about these automata are decidable and which are not, and there are tools for testing emptiness, evaluating reachable states etc. ([DOTY96]). However some theoretical aspects and parallels with ordinary finite automata are still not clear. This paper may be considered as a continuation of ([ACM97]) where timed languages were analyzed from the traditional linguistic viewpoint — and timed regular expression capable to specify exactly the same languages as timed automata were introduced.

We take for a model following classical (forty years old) results about finite automata, regular languages and linear equations (see e.g. [Brz62]).

Any system of linear equations in the form

$$X_i = \alpha_i + \sum_{j=1}^n \beta_{ij} X_j \quad i = 1, \dots, n, \quad (1)$$

where X_i stand for unknown languages and α_i, β_{ij} — for given regular coefficients, has a regular minimal solution. The regular expression for this solution can be found effectively from the coefficients.

For any finite automaton a system (1) can be easily constructed, each unknown X_i of the system corresponding to a state q_i of the automaton. In the

* This research was supported in part by the Russian Foundation for Basic Research under the grants 97-01-00692 and 96-15-96048; and by the International Association for the Promotion of Cooperation with Scientists from the Independent States of the Former Soviet Union (INTAS) under the grant 94-697.

minimal solution, the language X_i is exactly the language accepted by the automaton starting from the state q_i .

As a corollary these two classical results imply Kleene theorem ([Kle56]) about regularity of languages accepted by finite automata.

Our aim is to port these results to timed automata and to introduce a class of equations over timed languages capable to specify languages of one-clock timed automata. These equations are similar to classical linear equations (1). However the following example shows that a straightforward timed adaptation of linear equations cannot work.

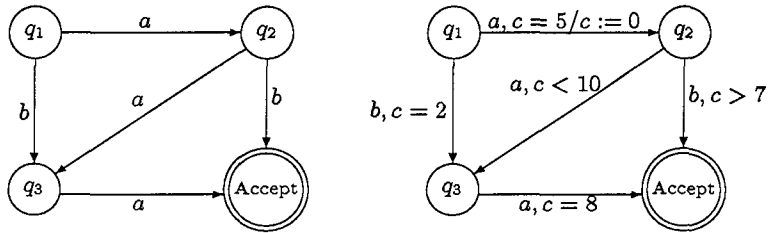


Fig. 1. Two automata

Example 1. The language of the first (untimed) automaton on Figure 1 can be represented by the following equations:

$$\begin{cases} X_1 = aX_2 + bX_3 \\ X_2 = b + aX_3 \\ X_3 = a \end{cases}$$

X_i here stands for the language accepted from the state q_i and each transition from q_i to q_j labeled with a can be represented by a term aX_j in the equation for X_i . Roughly speaking, such a transition corresponds to concatenating its label a to the language.

The case of the second (timed) automaton is more complicated because now there are two kinds of transitions. Some of them reset the clock and in this case they also can be represented by concatenation of the label (with time restriction) to the language. However some transitions do not reset the clock. We cannot write an equation like $X_1 = aX_2 + bX_3$ with a constraint on the sojourn time in state q_1 , because after completing action b the automaton enters the state q_3 with a modified clock value.

To deal with this problem we introduce another composition operation on timed languages (\circ operation) which corresponds to non-resetting transitions. We introduce quasilinear equations on timed languages which use both kinds

of concatenation (\cdot and \circ) and are strong enough to represent one-clock timed automata.

Our main result is that any system of equations of this class with regular coefficients has a regular minimal solution. We give an algorithm to find out this solution.

The paper is motivated by the theory of timed automata, however the major part of it (sections 3–4) contains an automata-free theory of timed languages, regular timed expressions and quasilinear equations on timed languages. At our opinion, this linguistic approach could be useful for other classes of hybrid systems as well.

The outline of the paper is as follows. In section 2 we recall the definition of timed regular languages from [ACM97]. In section 3 the new operation \circ over languages is formally introduced. This operation is crucial for representing timed automata by equations. We investigate algebraic properties of this operation and show, that \circ can be eliminated in a sense. In section 4 quasilinear equations are introduced and solved. The possibility to solve this kind of equations is the main result of the paper. In section 5 we recall the definition of timed automata and apply our main result to languages of these automata. For any one-clock automaton we construct a quasilinear system, which represents the language of this automaton. This provides an alternative proof of expressive equivalence of timed automata and timed regular expressions from ([ACM97]). In the last section further work is discussed.

2 Timed Regular Languages

We reproduce in a slightly modified form the basic definitions of timed languages and timed regular equations from [ACM97]. Let Σ be a finite *alphabet* and let \mathbb{R}_+ denote the set of positive real numbers. A *signal* over Σ is a timed sequence of elements of Σ , i.e. a finite sequence $w = ((a_1, t_1), \dots, (a_n, t_n))$ with $a_i \in \Sigma$ and $t_i \in \mathbb{R}_+$, such that $0 < t_1 < \dots < t_n$. We will also write this signal as

$$w = a_1^{r_1} a_2^{r_2} \dots a_n^{r_n},$$

where $r_1 = t_1$, and $r_{i+1} = t_{i+1} - t_i$, i.e. r_i are relative delays between a_i occurrences. We call t_n the *length* of w and denote it by $|w|$. The empty signal is denoted by ε . Its length equals 0. The set of all signals is denoted by $\mathcal{S}(\Sigma)$. Subsets of $\mathcal{S}(\Sigma)$ are referred to as *(timed) languages*. For every $w_1, w_2 \in \mathcal{S}(\Sigma)$ such that $w_1 = a_1^{r_1} a_2^{r_2} \dots a_n^{r_n}$ and $w_2 = b_1^{s_1} b_2^{s_2} \dots b_m^{s_m}$ we define their concatenation as $w = w_1 w_2 = a_1^{r_1} \dots a_n^{r_n} b_1^{s_1} \dots b_m^{s_m}$. This notion can be extended naturally to concatenation of languages by letting

$$L_1 L_2 = \{w_1 w_2 : w_1 \in L_1 \wedge w_2 \in L_2\}.$$

An *integer-bounded interval* is either $[l, u]$, $(l, u]$, $[l, u)$, or (l, u) where $l \in \mathbb{N}$ and $u \in \mathbb{N} \cup \{\infty\}$ such that $l \leq u$. We exclude ∞ and use l for $[l, l]$.

Definition 1 (Timed Regular Expressions). The set $\mathcal{E}(\Sigma)$ of timed regular expressions over an alphabet Σ , (expressions, for short) is defined recursively as either a , $\alpha_1 \cdot \alpha_2$, $\alpha_1 + \alpha_2$, α^* or $\langle \alpha \rangle_I$ where $a \in \Sigma$, $\alpha, \alpha_1, \alpha_2 \in \mathcal{E}(\Sigma)$ and I is an integer-bounded interval.

The semantics of timed regular expressions, $\llbracket \cdot \rrbracket : \mathcal{E}(\Sigma) \rightarrow \mathcal{E}^{S(\Sigma)}$, is given by:

$$\begin{aligned} \llbracket a \rrbracket &= \{a^r : r \in \mathbb{R}_+\} \\ \llbracket \alpha_1 + \alpha_2 \rrbracket &= \llbracket \alpha_1 \rrbracket \cup \llbracket \alpha_2 \rrbracket \\ \llbracket \alpha_1 \cdot \alpha_2 \rrbracket &= \llbracket \alpha_1 \rrbracket \llbracket \alpha_2 \rrbracket \\ \llbracket \alpha^* \rrbracket &= \bigcup_{i=0}^{\infty} (\llbracket \alpha^i \rrbracket) \\ \llbracket \langle \alpha \rangle_I \rrbracket &= \llbracket \alpha \rrbracket \cap \{w : |w| \in I\} \end{aligned}$$

Some comments should be given here. First, the semantics of a is not a singleton, but a non-countable language. The intuitive meaning of this expression is that some unknown time passes and then event a happens. Operations $+$, \cdot and $*$ are the same as for untimed languages. The only operation which introduces time explicitly is “time restriction” $\langle \cdot \rangle_I$ which chooses only those signals in the language, whose lengths belong to the constraining interval I .

Example 2.

$$\llbracket \langle \langle ab \rangle_{(2;3)} c \rangle_{100} \rrbracket = \{a^x b^y c^z \mid 2 < x + y < 3; x + y + z = 100\}.$$

To simplify notation we write ε for the following regular expression $\langle a^* \rangle_0$, whose semantics is exactly ε .

Expressions introduced here form a proper subclass of those introduced in [ACM97], because here intersection is not allowed in the syntax. This change explains the difference between the formulation of Theorem 15 below from that of the same theorem in [ACM97].

3 Operation \circ

Begin with the following *shift operation* over signals, which just delays the beginning by t and preserves relative delays between events.

Definition 2. For a signal $w = a_1^{t_1} a_2^{t_2} \dots a_n^{t_n}$ let $S^t w = a_1^{t_1+t} a_2^{t_2} \dots a_n^{t_n}$

We say that a language is *shift-invariant*, if $S^{-t}L = L$ for any $t > 0$, i.e. any signal w belongs (or does not belong) to L simultaneously with $S^t w$. The following condition is sufficient for shift invariance — the regular expression should not begin with something in $\langle \cdot \rangle$. Formally speaking

Lemma 3. *If a regular expression has a form $\sum_i \alpha_i \beta_i$ where $\alpha_i \not\equiv \varepsilon$ and α_i does not contain $\langle \cdot \rangle$, then its language is shift-invariant. We call this type of regular expressions dull.*

Now we can define a new composition operation over timed languages which is crucial for describing timed automata.

Definition 4. Let L_1 and L_2 be timed languages. Then

$$L_1 \circ L_2 = \{w_1 w_2 | w_1 \in L_1 \text{ and } S^{|w_1|} w_2 \in L_2\}.$$

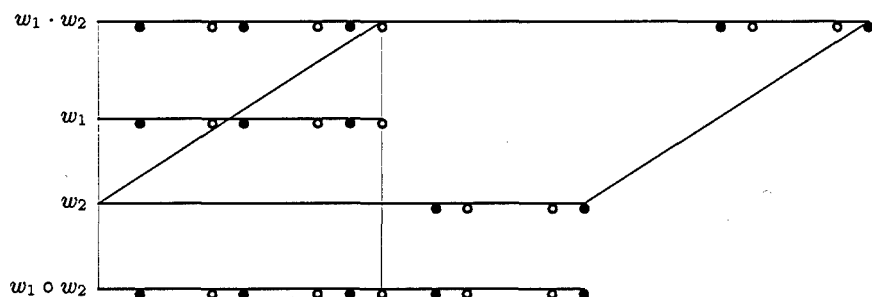


Fig. 2. Two compositions

In other words, for two signals $w_1 = ((a_1, t_1), \dots, (a_n, t_n)) \in L_1$ and $w_2 = ((b_1, s_1), \dots, (b_m, s_m)) \in L_2$ such that $t_n < s_1$ we include the signal $((a_1, t_1), \dots, (a_n, t_n), (b_1, s_1), \dots, (b_m, s_m))$ into $L_1 \circ L_2$. Figure 2 illustrates \circ -composition in comparison with concatenation.

First of all, state some simple algebraic properties of this composition operation.

Proposition 5 (Algebraic properties of circle). – operation \circ is $+$ -distributive: $(\alpha + \beta) \circ \gamma = \alpha \circ \gamma + \beta \circ \gamma$ and $\alpha \circ (\beta + \gamma) = \alpha \circ \beta + \alpha \circ \gamma$
 – operation \circ is associative: $(\alpha \circ \beta) \circ \gamma = \alpha \circ (\beta \circ \gamma)$
 – $\alpha \circ (\beta \gamma) = (\alpha \circ \beta) \gamma$ if $\varepsilon \notin \beta$

We suppose that \circ cannot be expressed in terms of other operations. However, it can be eliminated for regular languages.

Proposition 6 (Circle elimination). If L_1 and L_2 are regular, then $L_1 \circ L_2$ is regular. The regular expression for it can be obtained algorithmically.

Circle elimination is easy with the following prefix form of regular expressions

Lemma 7. Any regular expression can be effectively transformed to the form:

$$\gamma + \sum_{k=1}^n \langle \alpha_k \rangle_{I_k} \beta_k \quad (2)$$

or

$$\varepsilon + \gamma + \sum_{k=1}^n \langle \alpha_k \rangle_{I_k} \beta_k, \quad (3)$$

where γ is dull and $\alpha_k \not\preceq \varepsilon$.

The proof is by induction over the structure of regular expression. The only bad operation is Kleene star — all others are trivial. To deal with Kleene star suppose that δ is already in the prefix form (2) $\delta = (\gamma + \sum_k \langle \alpha_k \rangle_{I_k} \beta_k)$ and transform the expression δ^* to the form $\delta\delta^* + \varepsilon$ and open the parentheses:

$$\delta^* \delta\delta^* + \varepsilon = (\gamma + \sum_k \langle \alpha_k \rangle_{I_k} \beta_k) \delta^* + \varepsilon = \gamma\delta^* + \sum_k \langle \alpha_k \rangle_{I_k} \beta_k \delta^*.$$

which is already in the required form (3). The case when δ is in the form (3) is considered similarly.

It is easy to calculate \circ -composition with terms of (2) or (3):

- If γ is dull then $\delta \circ \gamma = \delta\gamma$;
- $\delta \circ \langle \alpha \rangle_I \beta = \langle \delta\alpha \rangle_I \beta$ if $\alpha \not\preceq \varepsilon$;
- $\delta\varepsilon = \delta$.

Proposition 6 is now immediate.

We illustrate Proposition 6 by the following example.

Example 3. Let us eliminate \circ from $\delta = \langle d \rangle_3 \circ (\langle ab \rangle_{8c})^*$. First transform the second term to the prefix form: $(\langle ab \rangle_{8c})^* = \langle ab \rangle_{8c} (\langle ab \rangle_{8c})^* + \varepsilon$, and second calculate $\delta = \langle \langle d \rangle_3 ab \rangle_{8c} (\langle ab \rangle_{8c})^* + \langle d \rangle_3$.

We can introduce the following analogue of Kleene star for \circ -composition.

Definition 8. $L^{\circ} = \varepsilon \cup L \cup L \circ L \cup L \circ L \circ L \cup \dots$

This operation can also be eliminated for regular languages. However this result is less straightforward.

Proposition 9 (Circled star elimination). *If L is regular, then L° is regular. The regular expression for it can be obtained algorithmically.*

Notice that this is easy for terms of (2). In fact, if γ is dull then $\gamma^{\circ} = \gamma^*$, and

$$(\langle \alpha \rangle_I \beta)^{\circ} = \langle \langle \alpha \rangle_I (\beta\alpha)^* \rangle_I \beta + \varepsilon.$$

The general case is more difficult. We give only a sketch of proof. First of all, transform the expression to the prefix form (2). Let $0 = \tau_0 < \tau_1 < \tau_2 < \dots < \tau_n = \infty$ be all the endpoints of intervals I_k . For each values of i and k either $(\tau_i, \tau_{i+1}) \subseteq I_k$ (in this case we say that α_k is *active* on (τ_i, τ_{i+1})), or $I_k \cap (\tau_i, \tau_{i+1}) = \emptyset$. If α_k is *active*, it means that it is allowed to terminate anywhere inside the interval (τ_i, τ_{i+1}) . Otherwise it is not allowed to terminate in (τ_i, τ_{i+1}) . Let $A_i = \{k | \alpha_k \text{ active on } (\tau_i, \tau_{i+1})\}$. γ is allowed everywhere, and β_k should happen after the corresponding active α_k . For each i we define a regular

expression $\mathcal{A}_i = (\gamma + \sum_{k \in A_i} \alpha_k \beta_k)^*$. Its language contains concatenations of words, active on (τ_i, τ_{i+1}) . Any word from \mathcal{A}_i if it fits into (τ_i, τ_{i+1}) may occur during this time interval.

Let w be a signal from L^\otimes . It can be parsed as follows:

$$w = w_0 \delta_0 w_1 \delta_1 \dots w_m \delta_m, \quad (4)$$

where $m \leq n$, τ_i occurs during w_i , shifts of w_i belong to some $\langle \alpha_k \rangle_{I_k} \beta_k$ or to γ and $\delta_i \in \mathcal{A}_i$. The idea behind this parsing is to see what happens at finitely many critical times τ_i and to allow any number of γ and $\alpha_k \beta_k$, where $k \in A_i$ to happen on the interval (τ_i, τ_{i+1}) (see Fig. 3).

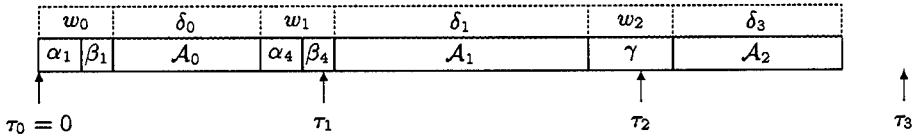


Fig. 3. Parsing a signal from L^\otimes

All these requirements can be written as a regular expression. For sake of simplicity we ignore the case when some w_i boundary is exactly at τ_i or if some w_i covers several consecutive τ_i .

For any τ_i find out what happens in w at τ_i , i.e. to which term $\langle \alpha_k \rangle_{I_k} \beta_k$ or to γ belongs (the shift of) w_i . We also find when τ_i occurs: during α or β . All this information for all the τ_i forms the *pattern* of the word w . Notice that there are finitely many possible patterns. An example pattern P (corresponding to Fig. 3) is as follows: " α_3 at τ_0 , β_4 at τ_1 , γ at τ_2 and the signal is finished before τ_3 " (for this pattern to be *valid*, α_3 and α_4 should be active at (τ_0, τ_1)). Now consider each pattern separately. For any pattern, using parsing (4) and expressions \mathcal{A}_i we can write a regular expression which defines the set of all the words in L^\otimes having this pattern. Instead of a heavy general formula consider only the expression corresponding to our sample pattern P:

$$\mathcal{E}_P = \langle \langle \langle \langle \alpha_1 \beta_1 \mathcal{A}_0 \alpha_4 \rangle_{(0, \tau_1)} \beta_4 \rangle_{(\tau_1, \tau_2)} \mathcal{A}_1 \rangle_{(\tau_1, \tau_2)} \gamma \rangle_{(\tau_2, \tau_3)} \mathcal{A}_2 \rangle_{(\tau_2, \tau_3)}.$$

Last, to obtain the final regular expression we sum expressions \mathcal{E}_P over all valid patterns P.

4 Quasilinear Equations

Definition 10. A system of quasilinear equations has the following form:

$$X_i = \alpha_i + \sum_{j=1}^n \beta_{ij} X_j + \sum_{j=1}^n \gamma_{ij} \circ X_j, \quad i = 1, \dots, n, \quad (5)$$

where X_i stand for unknown timed languages and $\alpha_i, \beta_{ij}, \gamma_{ij}$ — for given regular coefficients.

We can now formulate the main result of the paper.

Theorem 11. *The minimal solution of a system of quasilinear equations is regular. Its regular expression can be obtained algorithmically from expressions for the coefficients.*

The rest of this section is devoted to the sketch of proof of this theorem and algorithm description. Without loss of generality suppose that β_{ij} do not contain the empty signal ε . Otherwise we can move this empty signal from β_{ij} to γ_{ij} .

The first thing to do is to separate unknowns to which concatenation is applied from those to which \circ is applied. To achieve this aim we create another copy of each unknown.

Lemma 12. *The following system:*

$$\begin{cases} X_i = \alpha_i + \sum_{j=1}^n \beta_{ij} Y_j + \sum_{j=1}^n \gamma_{ij} \circ X_j, \\ Y_i = X_i \end{cases} \quad (6)$$

has the same solutions as the original system (5). Formally $X_1 = L_1, \dots, X_n = L_n$ is a solution to (5) iff $X_1 = Y_1 = L_1, \dots, X_n = Y_n = L_n$ is a solution to (6) and all the solutions to the latter system have this form.

The following lemma gives a solution to a single equation with only one operation. Its proof is fully similar to the proof of the same result for discrete equations.

Lemma 13. — *The minimal solution to $X = \alpha + \gamma \circ X$ is $X = \gamma^* \circ \alpha$;
— The minimal solution to $Y = \alpha + \beta Y$ is $Y = \beta^* \alpha$;*

The algorithm of solving the system (6) is similar to the classical algorithm for discrete languages and consists in iterated application of Lemma 13 together with circle elimination from Section 3. At the first stage we begin with the first equation and express X_1 from it as

$$X_1 = \gamma_{11}^* \circ (\alpha_1 + \sum_{j=1}^n \beta_{1j} Y_j + \sum_{j=2}^n \gamma_{1j} \circ X_j).$$

Eliminating circles, this equation can be transformed to the form

$$X_1 = \alpha'_1 + \sum_{j=1}^n \beta'_{1j} Y_j + \sum_{j=2}^n \gamma'_{1j} \circ X_j.$$

We put this expression into the second equation and solve it for X_2 . And we continue till X_n for which we find an expression that contains only Y s and not X

unknowns. Then the second stage begins. We go backwards putting this expression for X_n into equation number $n - 1$. This allows to find X_n -free expression for X_{n-1} and so on until we reach X_1 once again. Now the system has the form

$$\begin{cases} X_i = \alpha_i'' + \sum_{j=1}^n \beta_{ij}'' Y_j; \\ Y_i = X_i. \end{cases} \quad (7)$$

Replacing Y_i by X_i we obtain the o-free system

$$X_i = \alpha_i'' + \sum_{j=1}^n \beta_{ij}'' X_j;$$

and we express again

$$X_1 = \beta_{11}''^* (\alpha_1'' + \sum_{j=2}^n \beta_{1j}'' X_j),$$

put the result into the second equation, find X_2 and so on. This is the third stage of the algorithm. The fourth (and the last) stage consists in going backwards putting the regular expression for X_n into equation $n - 1$ and so on. This ends up with finding regular expressions for all the X_i . This concludes the algorithm and the proof of Theorem 11.

5 Applying Equations to Timed Automata

First recall shortly the definition of timed automata and their languages.

Definition 14 (Timed Automaton, [AD94]). A *timed automaton* is a tuple $\mathcal{A} = (Q, C, \Delta, \Sigma, S, F)$ where Q is a finite set of states, C is a finite state of clocks, Σ is an output alphabet, Δ is a transition relation (see below), $S \subseteq Q$ an initial set and $F \subseteq Q$ an accepting set. An element of the transition relation is of the form (q, ϕ, ρ, q', a) where q and q' are states, $a \in \Sigma$ -an output symbol, $\rho \subseteq C$ and ϕ (the transition guard) is a boolean combination of formulae of the form $(c \in I)$ for some clock c and some integer-bounded interval I .

A *clock valuation* is a function $\mathbf{v} : C \rightarrow \mathbb{R}_+ \cup \{0\}$ (which is the same a vector $\mathbf{v} \in (\mathbb{R}_+ \cup \{0\})^{|C|}$). We denote the set of all clock valuations by \mathcal{H} . For a clock valuation \mathbf{v} and a set $\rho \subseteq C$ we put for any clock variable $c \in C$

$$\text{Reset}_\rho \mathbf{v}(c) = \begin{cases} 0 & \text{if } c \in \rho \\ \mathbf{v}(c) & \text{if } c \notin \rho \end{cases}$$

That is, Reset_ρ resets to zero all the clocks in ρ and leaves the other clocks unchanged. We use $\mathbf{1}$ to denote the unit vector $(1, \dots, 1)$.

A *finite run* of the automaton is a sequence

$$(q_0, \mathbf{v}_0) \xrightarrow[t_1]{\delta_1} (q_1, \mathbf{v}_1) \xrightarrow[t_2]{\delta_2} \dots \xrightarrow[t_n]{\delta_n} (q_n, \mathbf{v}_n),$$

where $q_i \in Q$, $\mathbf{v}_i \in \mathcal{H}$, $\delta_i \in \Delta$, $t_i \in \mathbb{R}_+$, and which satisfies the following conditions:

Time progress: $0 < t_1 < \dots < t_n$ (for convenience we put $t_0 = 0$);

Succession: If $\delta_i = (q, \phi, \rho, q', a_i)$ then $q_{i-1} = q$, $q_i = q'$, the condition $\phi(\mathbf{v}_{i-1} + (t_i - t_{i-1})\mathbf{1})$ holds and $\mathbf{v}_i = \text{Reset}_\rho(\mathbf{v}_{i-1} + (t_i - t_{i-1})\mathbf{1})$.

An *accepting run* is a run satisfying the additional conditions:

Initialization: $q_0 \in S$; $\mathbf{v}_0 = \mathbf{0}$;

Termination: $q_n \in F$.

The *trace* of such a run is the signal

$$a_1^{t_1} a_2^{t_2 - t_1} \dots a_n^{t_n - t_{n-1}},$$

whose length is t_n . The *language of a timed automaton*, $L(\mathcal{A})$, consists of all the traces of its accepting runs.

Now recall the main result of [ACM97].

Theorem 15 [ACM97]. *A timed language L can be accepted by a timed automaton of Alur and Dill iff it can be represented in the form*

$$L = \varphi \left(\bigcap_{i=1}^n L_i \right),$$

where L_i are regular languages and φ — a homomorphism.

(The terminology of [ACM97] is slightly different.)

The difficult direction is of course to find regular expressions for a given automaton. This operation in ([ACM97]) is split into 2 parts. The first one consists in reduction to one-clock automata.

Lemma 16 [ACM97]. *Any timed language L accepted by a timed automaton can be represented in the form*

$$L = \varphi \left(\bigcap_{i=1}^n L_i \right),$$

where L_i are languages accepted by one-clock automata and φ — a homomorphism.

Our equation techniques is of no help here. However our result can simplify the proof of the second part.

Lemma 17 [ACM97]. *Any timed language L accepted by a one-clock timed automaton is regular.*

Given a one-clock timed automaton it is easy to construct an equivalent system of quasilinear equations.

In order to do it, for any control state of the automaton q_i introduce an unknown X_i . For a transition from q_i to the accepting state with the label a and the guard $(c \in I)$ put $\alpha_i = \langle a \rangle_I$. For a transition from q_i to q_j with label a , guard $(c \in I)$ and no reset put $\gamma_{ij} = \langle a \rangle_I$. For a transition q_i to q_j with label a , guard $(c \in I)$ and reset $(c := 0)$ put $\beta_{ij} = \langle a \rangle_I$. Finally write the system of equations

$$X_i = \alpha_i + \sum_{j=1}^n \beta_{ij} X_j + \sum_{j=1}^n \gamma_{ij} \circ X_j, \quad i = 1, \dots, n, \quad (8)$$

of the form (5).

The quasilinear system obtained in such a straightforward way from the one-clock automaton can be solved using the algorithm of the previous section. The following lemma concludes the new proof of Lemma 17 and Theorem 15.

Lemma 18. *X_i in the minimal solution of equations (8) is the language accepted by the automaton from the state q_i with initial value of the clock $c = 0$.*

Example 4. Consider the second (timed) automaton on the Figure 1. According to the general construction corresponding quasilinear equations are like this:

$$\begin{cases} X_1 = & \langle a \rangle_5 X_2 + \langle b \rangle_2 \circ X_3 \\ X_2 = \langle b \rangle_{(7,\infty)} & + \langle a \rangle_{(0,10)} \circ X_3 \\ X_3 = \langle a \rangle_8 \end{cases} \quad (9)$$

For the system 9 the procedure of section 4 gives the solution

$$\begin{cases} X_1 = \langle a \rangle_5 \langle b \rangle_{(7,\infty)} + \langle a \rangle_5 \langle \langle a \rangle_{(0,10)} a \rangle_8 + \langle \langle b \rangle_2 a \rangle_8 \\ X_2 = \langle b \rangle_{(7,\infty)} + \langle \langle a \rangle_{(0,10)} a \rangle_8 \\ X_3 = \langle a \rangle_8 \end{cases}$$

6 Conclusions and Further Work

In this paper a new linguistic formalism for timed languages is proposed. This formalism is adequate for timed automata. However there are still many questions to investigate.

- Which is the complexity of the algorithms?
- Is it possible to apply this approach directly to multi-clock timed automata?
- Which are other possible applications of this formalism? In particular, is it convenient for specification of timed systems?
- What can be done for more complicated equations?

References

- [ACM97] Eugene Asarin, Paul Caspi, and Oded Maler. A Kleene theorem for timed automata. In *Proc. 12th Annual IEEE Symposium on Logic in Computer Science*, pages 160–171, Warsaw, June 1997. IEEE Computer Society.
- [AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
- [Brz62] Janusz A. Brzozowski. A survey of regular expressions and their applications. *IRE Trans. on Electronic Computers*, EC-11(3):324–335, 1962.
- [DOTY96] Conrado Daws, Alfredo Olivero, Stavros Tripakis, and Sergio Yovine. The tool KRONOS. In Rajeev Alur, Thomas A. Henzinger, and Eduardo D. Sontag, editors, *Hybrid Systems III, Verification and Control*, number 1066 in Lecture Notes in Computer Science, pages 208–219. Springer-Verlag, 1996.
- [Kle56] S.C. Kleene. Representations of events in nerve nets and finite automata. In R. McNaughton and H. Yamada, editors, *Automata Studies*, pages 3–42. Princeton University Press, 1956.

Hybrid Control for Automotive Engine Management: The Cut-Off Case

A. Balluchi† M. Di Benedetto‡ C. Pinello† C. Rossi§ and

A. Sangiovanni-Vincentelli¶

†PARADES, Via San Pantaleo, 66, 00186 Roma, Italy,
balluchi,pinello,alberto@parades.rm.cnr.it

‡Dipartimento di Ingegneria Elettrica, Università dell'Aquila, Poggio di Roio, 67040
L'Aquila, Italy, dibenede@giannutri.caspur.it

§Magneti Marelli S.p.A., Via del Timavo, 33, 40134 Bologna, Italy,
Carlo.Rossi@bologna.marelli.it

¶Department of Electrical Engineering and Computer Science, University of
California at Berkeley, CA 94720

Abstract. A novel approach to the control of an automotive engine in the cut-off region is presented. First, a hybrid model which describes the torque generation mechanism and the power-train dynamics is developed. Then, the cut-off control problem is formulated as a hybrid optimization problem, whose solution is obtained by relaxing it to the continuous domain and mapping its solution back into the hybrid domain. A formal analysis as well as simulation results demonstrate the properties and the quality of the control law.

1 Introduction

Hybrid systems have been the subject of intensive study in the past few years with particular emphasis placed on a unified representation of the problem in terms of rigorous mathematical foundations (see [5], [10], [7], [6], [3], [4]). In our opinion, it is important to address significant domains of application of hybrid control to develop further understanding of the implications of the model on the control algorithms. In the automotive industry, engine behavior is in general partitioned into regions of operation where appropriate control action are applied to yield the desired result [2]. The region of operation considered here is characterized by the driver who, by releasing the gas pedal, requests no torque to the engine. An obvious strategy to minimize gas consumption and emissions when no torque is requested is to shut fuel injection off, an operation called *cut-off*. However, cutting off fuel injection as soon as the gas pedal is released, causes a sudden torque reduction that may result in unpleasant oscillations compromising driving comfort. The open-loop control policy implemented by the industry today, consists of air and fuel input modulation, that is, throttle closure is slowed down and, when air quantity is below a threshold, fuel injection is gradually reduced to zero. As is often the case, heuristic rule-based controls need extensive tuning, yield satisfactory solutions only in a limited range of operations and are hardly optimal with respect to the emissions and fuel consumption.

In this paper, we introduce a novel, theoretically sound, closed-loop approach to cut-off control. The "plant" consists of two parts: the engine responsible for torque generation, modeled as a combination of an Extended Finite State Machine and of a Discrete Event system, and the powertrain modeled as a fourth-order linear Continuous Time system. The goal is to control the evolution of the system from an initial condition (the state of the system when the gas pedal has been released and the manifold pressure has approached its idle regime value) to cut-off, minimizing the amplitude of the undesired oscillations. The available control actions are on fuel injection and spark ignition, the inputs to the engine, occurring only once per engine cycle for each piston. The torque generated is the output of the engine and the input to the powertrain. The powertrain dynamics contain the potential oscillatory behavior to be minimized. The wheel revolution speed and the angular velocity of the crankshaft are components of its state. The timing of the torque generation mechanism is determined by the angle of the crankshaft. Consequently, the control problem is a Hybrid System Control problem, which is further complicated by the delay between the time in which the decision on the quantity of fuel to be injected is taken (at the beginning of the exhaust phase) and the time the effect of this decision takes place (during the next expansion phase).

Our approach to the hybrid problem at hand is to relax the problem to the continuous domain assuming that the torque signal can be modulated continuously over time within a given range of values. The problem so obtained is non trivial since the objective function is non differentiable. In [1] we devised a strategy that solves this problem to yield an optimal control law for the continuous domain. Then, this solution in the continuous domain was mapped back into the discrete domain. The control algorithm used sliding mode control in a region of the state space. Sliding modes yielded a control strategy that had implementation problems since it is required a fairly large number of switchings in the injection policy. In this paper we present a novel strategy which eliminates completely the sliding mode. The solution, mapped back in the hybrid domain, is demonstrated to yield a behavior that is close (within a precisely specified bound) to the behavior of the control in the continuous case. From the application of our control strategy by our industrial partner on a commercial vehicle, the proposed solution appears to be far superior than the present, heuristic approach in performance, emission control, memory and CPU occupation.

2 Problem formulation

2.1 Plant model

In this paper, we deal with 4-stroke N -cylinder gasoline engines. Our model consists of the composition of N sub-models, one per cylinder. The single cylinder sub-model \mathcal{M} consists of three parts:

¹ A deterministic Finite State Machine is a six-tuple $M = \{I, Y, S, s_0, \lambda, \gamma\}$ where I is the (finite) set of inputs, Y is the (finite) set of outputs, S is the finite set of states, s_0 is the initial state, $\lambda : S \times I \rightarrow S$ is the next state function, $\gamma : S \times I \rightarrow Y$ is the output function. An Extended Finite State Machine is an FSM where the input and output space are not necessarily finite and can be subsets of \mathbb{R}^n .

1. an Extended Finite State Machine (EFSM)¹ describing pistons' behavior;
2. a Discrete Event system (DE)² modeling torque generation;
3. a Continuous Time system (CT) modeling the powertrain.

Powertrain Model. The powertrain model is described by a continuous time system model developed at Magneti Marelli Engine Control Division. The model, whose parameters have been identified and validated, contains phenomena involved in powertrain oscillations that are of interest in cut-off control. Powertrain dynamics are modeled by the linear system

$$\dot{\zeta} = A^p \zeta + b^p u \quad (1)$$

$$\dot{\phi}_c = \omega_c \quad (2)$$

State $\zeta = [\alpha_e, \omega_c, \omega_p]^T$ represents the axle torsion angle, the crankshaft revolution speed, the wheel revolution speed, and ϕ_c represents the crankshaft angle. The input signal u is the torque acting on the crank. The linearized powertrain dynamics (1) is asymptotically stable since it models a passive mechanical system, and is characterized by a real dominant pole λ_1 , and a pair of conjugate complex poles $\lambda \pm j\mu$, responsible for the oscillating behavior.

Piston's behavior. The behavior of each piston in the engine is abstractly represented by the Extended Finite State Machine shown in Figure 1, where $S = \{H, I, C, E\}$. The four states of the EFSM are as follows.

- *Exhaust run (H).* The piston goes up, expelling combustion exhaust gases.
- *Intake run (I).* During its down-run the piston loads the air-fuel mix.
- *Compression run (C).* During its up movement the piston compresses the loaded mix.
- *Expansion run (E).* The compressed mix combustion, generated by a spark signal, produces a sudden pressure increase which pushes the piston downwards.

The transitions of the EFSM occur when a piston reaches the bottom or top dead point. The guard condition enabling the transition is expressed in terms of the piston position $\tilde{\phi}$ measured on the crankshaft, considering the offset ϕ_{co} which corresponds to the angle the crank is mounted on the shaft. The EFSM outputs the integer variable k which is incremented by one at each transition.

Torque generation. The quantity of air entering the cylinders during the intake run is controlled by a throttle valve (often directly connected to the gas pedal). The control system keeps the fuel quantity proportional to air load, so that the combustion process produces a minimum amount of waste and is maximally

² A DE system is intended in the sense of [8].

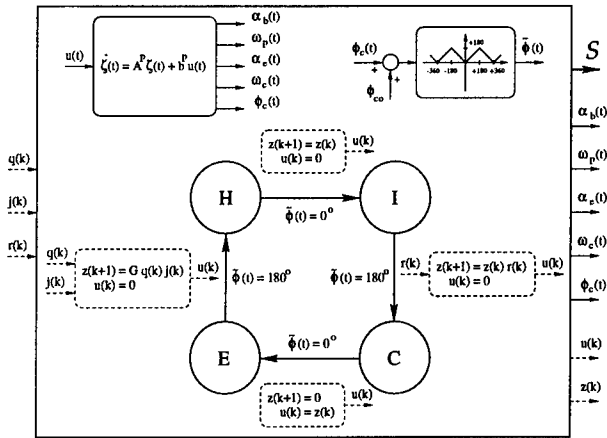


Fig. 1. Hybrid model for a single cylinder.

efficient. In electronic injection controlled engines, the fuel quantity is determined by the duration of the injection phase that takes place during the exhaust run.

The torque is generated during the expansion phase. Ideally, spark timing should occur at the precise time the piston reaches the end of its up-run in the compression phase. However, since combustion takes a non zero time to complete, it is convenient to time spark ignition before the piston concludes the compression phase. The time for spark ignition is usually referred to as *spark advance*, and it is expressed in terms of the angle the crankshaft covers before the piston reaches its next top dead center. Spark advance is not constant. It depends on temperature of the cylinders and of the loaded mix, and on the revolution speed of the crankshaft. The effect of spark advance, the other control variable considered here, is a modulation of the maximum value of torque that can be generated given the quantity and quality of the combustion mix. If the spark advance is optimal for torque generation, the modulation factor has the value 1, otherwise it is positive but less than one.

The generated torque is a complex function of time during the expansion phase, in practice it is replaced by the average value over the interval of time corresponding to the expansion phase.

The process of torque generation is characterized by the delays between the times in which fuel injection and spark advance are set and the time in which such decisions have an effect. Control signals are then subject to a transport process which can be represented by a DE system which is active at every EFSM transition. Such DE system, represented in Figure 1 with dashed boxes, receives as inputs:

- the integer k (an output of the EFSM);
- the mass of air-fuel mix $q \in \mathbb{R}^+$ loaded in the intake phase;

- a binary control variable $j \in \{0, 1\}$ which indicates whether or not the fuel is present in the mix;
- the modulation factor $r \in [r_{min}, 1]$ due to non optimal spark timing.

The DE system output is the torque $u(k)$. At the EFSM transition $E \rightarrow H$ the DE system reads its inputs $q(k)$ and $j(k)$, and stores in its state $z \in \mathbb{R}$ the maximum amount of torque achievable during the next E phase, obtained by the mix-to-torque gain G . Such value is corrected at the $I \rightarrow C$ transition by the modulation factor $r(k)$ due to the chosen spark advance. The DE output $u(k)$ is always zero except at the $C \rightarrow E$ transition when it is set to the value stored in z . Between two transitions of the EFSM, occurring at times t_k and t_{k+1} , the input signal $u(t)$ to the powertrain model is given by $u(t) = u(k)$ for $t \in [t_k, t_{k+1})$.

Engine hybrid model. An engine is characterized by the number of cylinders, most cars have four, but there are engines that have a different number of cylinders³. The pistons are connected to the crankshaft, so that the phases of their behavior are related to each other. The overall model of torque generation for a N -cylinder engine is then the combination of N EFSMs as in Figure 1 and of N DE systems representing the behavior of each piston. The hybrid model of the complete engine is obtained by adding to torque generation model the powertrain CT dynamics (1).

In this paper we focus on the most relevant case of a 4-cylinder engine. Its model, referred to in the rest of the paper as \mathcal{M}_{4cyl} , has input signals $\mathbf{j} = [j_1, j_2, j_3, j_4]^T$ and $\mathbf{r} = [r_1, r_2, r_3, r_4]^T$ properly synchronized with the corresponding DE models; we denote by \mathcal{J}_{4cyl} and \mathcal{R}_{4cyl} the classes of functions $\mathbb{N} \rightarrow \{0, 1\}^4$ and $\mathbb{N} \rightarrow [r_{min}, 1]^4$, feasible for \mathbf{j} and \mathbf{r} . Signal q is instead shared among the cylinders. Without loss of generality, we assume that in \mathcal{M}_{4cyl}

- the initial EFSM states are $S_{01} = H$, $S_{02} = I$, $S_{03} = C$, $S_{04} = E$;
- the crankshaft offsets ϕ_{co_i} are $\phi_{co_1} = \phi_{co_3} = 180^\circ$, $\phi_{co_2} = \phi_{co_4} = 0^\circ$;
- the initial value of the crankshaft angle $\phi_c(0)$ is set to zero.

2.2 The optimization problem

The optimization problem is to control fuel injection and spark advance so that vehicle acceleration peaks are minimized during the cut-off operation. Throttle closure produces a decreasing evolution of the manifold pressure towards the idle regime value. We identify as the starting point of the cut-off operation the time $t_0 = t_{\bar{k}}$ at which $q_a(\bar{k})$ equals the steady-state air quantity with pedal released, q_a^o . We also assume that before t_0 the injection signals were active, so that $z_1(0) = z_2(0) = z_3(0) = z_4(0) = Gq_a^o$. To simplify notation, we set $t_0 = 0$ and $\bar{k} = 0$.

³ For example, Formula 1 racing cars can have 8, 10 or 12 cylinders. Fiat Coupè 2000 Turbo has 5.

Assuming vehicle speed equal to wheel speed, vehicle acceleration is $a(t) = R \dot{\omega}_p(t)$, where R is the wheel radius. To isolate oscillations from monotone behavior, the following state transformation is applied. Set

$$\begin{bmatrix} x' \\ x \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} \zeta = P \zeta \quad (3)$$

with $x' \in \mathbb{R}$, $x \in \mathbb{R}^2$, $P_1 \in \mathbb{R}^{1 \times 3}$, and $P_2 \in \mathbb{R}^{2 \times 3}$, where P is obtained from the eigenvectors of A^p . Rewrite (1) as

$$\begin{bmatrix} \dot{x}' \\ \dot{x} \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & A \end{bmatrix} \begin{bmatrix} x' \\ x \end{bmatrix} + \begin{bmatrix} b' \\ b \end{bmatrix} u, \quad \text{where } A = \begin{bmatrix} \lambda & -\mu \\ \mu & \lambda \end{bmatrix}. \quad (4)$$

Denoting by $c \in \mathbb{R}^{1 \times 2}$ the product between the third row of A^p , the last two columns of P^{-1} and R , the oscillating component of the acceleration can be expressed as an output of the following linear system in the reduced state x

$$\dot{x}(t) = Ax(t) + bu(t) \quad (5)$$

$$\tilde{a}(t) = R \dot{\omega}_p(t) = c x(t). \quad (6)$$

The objective of the cut-off control strategy is to minimize the peaks of the acceleration $\tilde{a}(t)$ until they are less than a threshold of perception $\tilde{a}_{th} > 0$.

Consider the circle

$$B_\rho = \{x \in \mathbb{R}^2 : \|x\| \leq \rho, \quad \rho = \tilde{a}_{th} \|c\|^{-1}\}, \quad (7)$$

where $\|\cdot\|$ denotes the Euclidean norm. By asymptotic stability of system (5), when $u(t) = 0$ the norm of $x(t)$ decreases over time since $\frac{d(x^T x)}{dt} = 2\lambda x^T x$ and $\lambda < 0$. Therefore, if at some time \bar{t} , the state has been driven to $x(\bar{t}) \in B_\rho$, the control $u(t) = 0$ keeps the trajectory in B_ρ and the acceleration $\tilde{a}(t)$ is bounded above by the acceleration \tilde{a}_{th} for $t \geq \bar{t}$. Cut-off control problem can then be formulated as the optimization problem of steering the state x to the circle B_ρ minimizing the acceleration peak. Once in B_ρ , fuel injection can be safely shut off with vehicle oscillations below threshold.

Then the cut-off optimization problem can be formulated as follows.

Problem 1. Given the engine hybrid model \mathcal{M}_{4cyl} , find $\hat{j} \in \mathcal{J}_{4cyl}$ and $\hat{r} \in \mathcal{R}_{4cyl}$ such that

$$\sup_{0 \leq t \leq T} |\tilde{a}(t)| \Big|_{\substack{j = \hat{j} \\ r = \hat{r}}} = \min_{\substack{j \in \mathcal{J}_{4cyl} \\ r \in \mathcal{R}_{4cyl}}} \sup_{0 \leq t \leq T} |\tilde{a}(t)| \quad (8)$$

$$\text{subject to: } \begin{cases} \text{Dynamics of Hybrid Model } \mathcal{M}_{4cyl} \text{ with } \zeta(0) = \zeta_0 \text{ s.t.} \\ x(0) = x_0 \notin B_\rho, \\ x(T) \in B_\rho, \\ z_1(0) = z_2(0) = z_3(0) = z_4(0) = Gq_a^o, \\ q(k) = q_a^o, \text{ for all } k \geq 0 \end{cases} \quad (9)$$

where $x_0 = P_2 \zeta_0$, $x(T) = P_2 \zeta(T)$, \tilde{a} is given by (6), the final time T is free, $(\zeta_0, 0)^T$ is the continuous state value at the beginning of the cut-off operation, B_ρ is as in (7) and q_a^o is the steady-state air quantity with gas pedal released.

3 Continuous-time model solution

The main difficulties in Problem 1 are that the plant to be controlled is hybrid and that the input signals are bounded. Our strategy is to relax first the hybrid problem into the continuous domain, and then to map the solution back to the hybrid domain. The relaxed problem is as follows:

Problem 2. Given

$$\mathcal{U} = \{u : [0, +\infty) \rightarrow \mathbb{R} \mid u(t) \text{ is measurable and } 0 \leq u(t) \leq M, \forall t \geq 0\} \quad (10)$$

with $M = Gq_a^o$ and q_a^o the steady-state air quantity after pedal release,

$$\min_{u \in \mathcal{U}} \sup_{0 \leq t \leq T} |\tilde{a}(t)| \quad (11)$$

$$\text{subject to: } \begin{cases} \dot{x}(t) = Ax(t) + bu(t) \\ x(0) = x_0 \notin B_\rho \\ \|x(T)\| = \rho \end{cases} \quad (12)$$

where $|\tilde{a}(\cdot)|$ is as in (6) and T is finite.

Let $R(\theta) \in \mathbb{R}^2$ be the θ rotation matrix in \mathbb{R}^2 and let $(\cdot)_\perp$ be the $+\pi/2$ rotation operator defined as $z_\perp = R(\frac{\pi}{2})z$ for any $z \in \mathbb{R}^2$. Let $x_M = -A^{-1}bM$ be the equilibrium point with $u = M$ and $v = -\|x_M\|^{-1}(x_M)_\perp$. In our previous paper [1], it was shown that there exists $u \in \mathcal{U}$, which steers the state of (5) to the origin along a straight line, provided that x_0 is inside the domain $\mathcal{D}_M = \mathcal{C}_M/\mathcal{A}_v$, where $\mathcal{A}_v = \{x \in \mathbb{R}^2 \mid (v^T x)(b^T R(-\frac{\pi}{2})x) > 0\}$ and $\mathcal{C}_M = \{x \in \mathbb{R}^2 \mid x^T R(-\frac{\pi}{2})(Ax + bM) \leq 0\}$ (see⁴Figure 2). During such motion the cost function $|\tilde{a}(\cdot)|$ monotonically decreases to zero. However, the time to reach B_ρ becomes unbounded when x_0 is such that $v^T x_0 \rightarrow 0^-$.

Proposition 3. Let $\tilde{A} = \inf_{u \in \mathcal{U}} \sup_{0 \leq t \leq T} |\tilde{a}(t)|$ subject to (5). Consider

$$\bar{u} = \begin{cases} \begin{cases} 0 & \text{if } v^T x \geq 0 \\ M & \text{if } v^T x < 0 \end{cases} & \text{if } x \notin \mathcal{D}_M \\ \begin{cases} -\frac{x^T R(-\frac{\pi}{2})Ax}{x^T R(-\frac{\pi}{2})b} & \text{if } (R(\theta)v)^T x \geq 0 \\ 0 & \text{if } (R(\theta)v)^T x < 0 \end{cases} & \text{if } x \in \mathcal{D}_M \end{cases} \quad (13)$$

If $(cx_M)(cb) > 0$ then (13), with $\theta \in (0, \cos^{-1}(|b^T v||b|^{-1}))$ is an optimal solution to Problem 2, i.e. $\tilde{A} = \sup_{0 \leq t \leq T} |\tilde{a}(t)|$ with u as in (13). Otherwise, no optimal solution in finite time exist, but, for any $\epsilon > 0$ exists $\theta > 0$ such that control (13) steers x_0 to B_ρ in finite time with $\sup_{0 \leq t \leq T} |\tilde{a}(t)| - \tilde{A} < \epsilon$.

For all models of existing cars available to us, $(cx_M)(cb) > 0$ was verified so (13) was actually optimal. Figure 2 reports the closed loop phase space under control (13) with $(cx_M)(cb) > 0$ and $\theta = 0$. Let $\psi(u, x_0, t)$ denote a trajectory of (5)

⁴ \mathcal{C}_M has center in $\frac{M}{2\mu} R(\frac{\pi}{2})b$ and radius $\frac{M}{2\mu} \|b\|$. Its boundary contains the origin (with tangent collinear to vector b) and $x_M = -A^{-1}bM$, the equilibrium point with $u = M$.

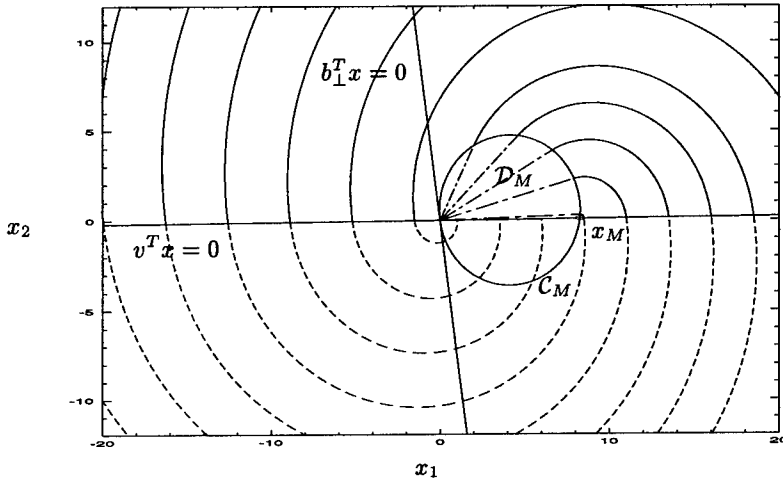


Fig. 2. Optimal trajectories for the relaxed control problem. If $x \notin \mathcal{D}_M$, u is equal to either M (continuous line) or 0 (dotted line). Otherwise $0 < u < M$ (dash-dot line).

and let $\mathcal{R}(\tau, x_0) = \{z \in \mathbb{R}^2 | \exists u \in \mathcal{U}, \text{ s.t. } z = \psi(u, x_0, \tau)\}$ be the reachable set in a specified time τ from an initial point x_0 . We shall prove Proposition 3 analyzing the properties of $\mathcal{R}(\tau, x_0)$. Direct application of basic elements of trajectory analysis gives the following results.

Proposition 4. (Proposition 5.1.1 in [9]). Given system (5) with $x(0) = x_0$ and $u \in \mathcal{U}$, the reachable set $\mathcal{R}(\tau, x_0)$ is convex and compact.

Proposition 5. (Theorem 8.1.1 in [9]). Given system (5) with $x(0) = x_0$ and $u \in \mathcal{U}$, if $x \in \mathcal{R}(\tau, x_0)$ then x can be reached from x_0 in time τ by means of a trajectory that corresponds to a bang-bang control with a finite number of switchings, that is $u(t) \in \{0, M\}$ for any $t \in [0, \tau]$ and $u(t)$ has finite number of discontinuity points in $[0, \tau]$.

Proposition 6. (Corollary 2.4.3 in [9]). Let $\psi(u, x_0, t)$ be a trajectory of (5) that corresponds to the feasible input $\hat{u}(\cdot) \in \mathcal{U}$, from x_0 to $x_\tau = \psi(u, x_0, \tau)$. Suppose that x_τ is not an interior point of $\mathcal{R}(\tau, x_0)$. Introduce the adjoint variables $p \in \mathbb{R}^2$ and Hamiltonian $H(\psi, p, \hat{u}) = p^T(Ax + b\hat{u})$. Then, there exists a trajectory $p(\cdot)$ such that $\dot{p} = -\frac{\partial H}{\partial x} = -A^T p$ with $p(t) \neq 0$ for all $t \in [0, \tau]$ and

$$\min_{u \in \mathcal{U}} H(\psi(u, x_0, t), p(t), u(t)) = H(\psi(\hat{u}, x_0, t), p(t), \hat{u}(t)) = \nu \quad (14)$$

for almost all $t \in [0, \tau]$ with ν a constant.

Corollary 7. Let $\psi(\hat{u}, x_0, t), p(t), \hat{u}(t)$ satisfy the minimum condition (14), then $\hat{u}(t) = M$, if $b^T p(t) > 0$, and $\hat{u}(t) = 0$, if $b^T p(t) < 0$. For any time interval of length less than or equal to $\frac{\pi}{\mu}$, $\hat{u}(t)$ has at most one discontinuity point.

Proposition 8. Let $\psi(\bar{u}, x_0, t)$ be the solution of (5), from x_0 , with \bar{u} as in (13).

(1) If $\tau > 0$ is a point of local maximum for $|\bar{a}(t)|$ and $\bar{u}(t)$ is continuous at $t = \tau$, then $\bar{u}(\tau) \in \{0, M\}$ and $x_\tau = \psi(\bar{u}, x_0, \tau)$ is such that $cA(x_\tau - x_M \frac{\bar{u}(\tau)}{M}) = 0$.

(2) If $\tau_1, \tau_2 > 0$ are two successive local maxima for $|\bar{a}(t)|$, then $\tau_2 - \tau_1 \leq \pi/\mu$ (where equality holds if $\bar{u}(t)$ is continuous at τ_1 and τ_2), and $|\bar{a}(\tau_1)| > |\bar{a}(\tau_2)|$.

Using Proposition 8, we can concentrate only on reachable sets in time $\tau \leq \frac{\pi}{\mu}$ to prove optimality of (13).

Corollary 9. Given system (5) and input class \mathcal{U} , points on the boundary of the reachable set $\mathcal{R}(\tau, x_0)$, with $\tau \leq \frac{\pi}{\mu}$, are reached from x_0 by means of a bang-bang control $u : [0, \tau] \rightarrow \{0, M\}$ with at most one switching.

The boundary of the reachable set $\mathcal{R}(\tau, x_0)$ for $\tau \leq \frac{\pi}{\mu}$, is readily obtained as

$$\partial\mathcal{R}(\tau, x_0) = \{x \in \mathbb{R}^2 \mid x = \gamma_0(\alpha) \vee x = \gamma_M(\alpha), \text{ with } \alpha \in [0, \tau]\},$$

where $\gamma_0(\cdot), \gamma_M(\cdot)$ are the parametric curves

$$\gamma_0(\alpha) = e^{A\tau}x_0 + (I - e^{A(\tau-\alpha)})x_M, \quad \gamma_M(\alpha) = e^{A\tau}x_0 + (e^{A(\tau-\alpha)} - e^{A\tau})x_M$$

corresponding to $u(t) = 0$ ($u(t) = M$, resp.) for $t \in [0, \alpha]$ and $u(t) = M$ ($u(t) = 0$, resp.) for $t \in [\alpha, \tau]$. Curves $\gamma_0(\cdot), \gamma_M(\cdot)$ are of class C^∞ and $\partial\mathcal{R}(\tau, x_0)$ is closed for $\gamma_0(\tau) = \gamma_M(0) = e^{A\tau}x_0 = x^{(0)}$ and $\gamma_M(\tau) = \gamma_0(0) = e^{A\tau}(x_0 - x_M) + x_M = x^{(M)}$. However, curve $\partial\mathcal{R}(\tau, x_0)$ is not, in general, of class C^1 at $x^{(0)}$ and $x^{(M)}$, since

$$\frac{\partial\gamma_0}{\partial\alpha} = Ae^{A(\tau-\alpha)}x_M \text{ and } \frac{\partial\gamma_M}{\partial\alpha} = -Ae^{A(\tau-\alpha)}x_M. \quad (15)$$

Proof of Proposition 3. We shall first consider the case $\theta = 0$ in (13) and discuss later the introduction of a $\theta \neq 0$. Let $\psi(\bar{u}, x_0, t)$ be a trajectory of system (5) corresponding to control \bar{u} as in (13), with $\theta = 0$, from an initial condition x_0 . Note that, for any t such that $\psi(\bar{u}, x_0, t) \notin \mathcal{D}_M$, $\psi(\cdot)$ is of class C^1 .

For particular x_0 , $|\bar{a}(\cdot)|$ is monotonic along $\psi(\bar{u}, x_0, t)$, from x_0 to $(0, 0)^T$, and hence achieves its maximum at x_0 . One can easily obtain that, the region \mathcal{Q} of such initial conditions contains \mathcal{D}_M and is bounded as follows: if $(cx_M)(cb) > 0$, by lines $cA(x - x_M) = 0$, $cAx = 0$, and the two trajectories with $u = M$ passing resp. through $(0, 0)^T$ and x_σ , with x_σ s.t. $cA(x_\sigma - x_M) = 0$ and $cx_\sigma = 0$; otherwise, if $(cx_M)(cb) \leq 0$, by lines $cA(x - x_M) = 0$, $v^T x = 0$, and the trajectory with $u = M$ passing through $(0, 0)^T$.

If $x_0 \notin \mathcal{Q}$, $|\bar{a}(\cdot)|$ achieves a maximum along $\psi(\bar{u}, x_0, t)$, for some time $\tau > 0$ such that $x_\tau = \psi(\bar{u}, x_0, \tau)$ satisfy either $cA(x_\tau - x_M) = 0$ or $cAx_\tau = 0$. Optimality of control (13) is proved by showing that at time τ , any other control $u \in \mathcal{U}$, different from \bar{u} , achieves a value of $|\bar{a}(\cdot)|$ which cannot be lower than $|c\psi(\bar{u}, x_0, \tau)|$:

$$|c\psi(\bar{u}, x_0, \tau)| \leq |c\psi(u, x_0, \tau)| \leq \sup_{0 \leq t \leq T} |c\psi(u, x_0, t)| \text{ for any } u \neq \bar{u} \in \mathcal{U}. \quad (16)$$

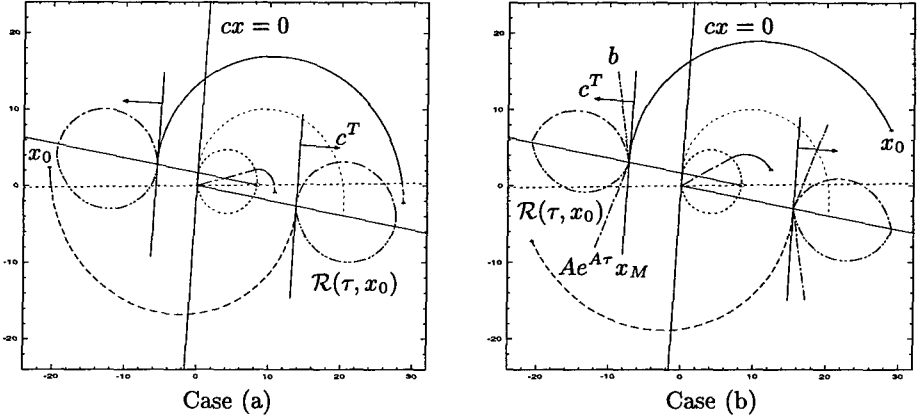


Fig. 3. Reachable sets for final conditions on the locus of points maximum.

Such inequality will be proved analyzing the reachable set $\mathcal{R}(\tau, x_0)$. From Proposition 8, we have $\tau \leq \frac{\pi}{\mu}$. Consider $x_0 \notin \mathcal{D}_M$. Two cases are in order either: (a) $\bar{u}(t)$ has a discontinuity point for some $\bar{\tau} \in [0, \tau]$, or (b) $\bar{u}(t)$ is constant in $[0, \tau]$.

(a) We first show that, in this case, $\frac{dx}{dt}(\tau)$ is collinear to the line s passing through x_τ and parallel to $Ae^{A(\tau-\bar{\tau})}x_M$. Consider $\bar{u}(t) = 0$ for $t \in [0, \bar{\tau}]$ and $\bar{u}(t) = M$ for $t \in [\bar{\tau}, \tau]$. We have $\frac{dx}{dt}(\tau) = Ae^{A\tau}(x_0 - e^{-A\bar{\tau}}x_M)$. But, since from (13) the operator $e^{A\bar{\tau}}$ steers point x_0 to the switching line $v^Tx = 0$, $e^{-A\bar{\tau}}$ maps x_M (which lies on $v^Tx = 0$) to a vector collinear to x_0 . Then, $\frac{dx}{dt}(\tau)$ is parallel to $Ae^{A(\tau-\bar{\tau})}x_M$. If, instead, $\bar{u}(t) = M$ for $t \in [0, \bar{\tau}]$ and $\bar{u}(t) = 0$ for $t \in [\bar{\tau}, \tau]$, $\frac{dx}{dt}(\tau) = Ae^{A\tau}((x_0 - x_M) + e^{-A\bar{\tau}}x_M)$ and, analogously to above, $e^{-A\bar{\tau}}$ maps x_M to a vector collinear to $x_0 - x_M$. Hence, in both cases $\frac{dx}{dt}(\tau)$ is collinear to s . Furthermore, from (15) one can easily obtain that line s is also tangent to $\partial\mathcal{R}(\tau, x_0)$ at x_τ . Due to the convexity property of the reachable set, $\mathcal{R}(\tau, x_0)/x_\tau$ is all located on the same side of s , namely the one which does not contain the origin. Since these arguments are valid for any final time, it follows that $\mathcal{R}(t', x_0)$, with $t' \leq \frac{\pi}{\mu}$, is always tangent to $\psi(\bar{u}, x_0, t)$ at $t = t'$.

Now, since by construction, $|\bar{a}(\cdot)|$ is maximal along $\psi(\bar{u}, x_0, t)$ at $t = \tau$, c is perpendicular to $Ae^{A(\tau-\bar{\tau})}x_M$ and s can be written as $(cx_\tau)c(x - x_\tau) = 0$. Since $\forall x \in \mathcal{R}(\tau, x_0)/x_\tau$, $(cx_\tau)c(x - x_\tau) > 0$, condition (16) holds for any $u \neq \bar{u}$.

(b) If instead $\bar{u}(t)$ is constant in $[0, \tau]$, necessarily, x_τ equals either $x^{(0)}$ or $x^{(M)}$ and, hence, $\partial\mathcal{R}(\tau, x_0)$ is not of class C^1 at x_τ . Evaluating from (15) the tangent vectors to $\gamma_0(\cdot)$ or $\gamma_M(\cdot)$ at x_τ , one obtains a direction parallel to b (with $\alpha = \tau$) and a direction parallel to $Ae^{A\tau}x_M$ (with $\alpha = 0$). Further, it is easy to check that, for the same x_τ , the latter ranges from the line parallel to b (as $x_0 \rightarrow x_\tau$) and the line perpendicular to c^T (as $v^Tx_0 \rightarrow 0$). Vector b is not collinear to $\frac{dx}{dt}(\tau)$ except for x_τ lying on $v^Tx = 0$. Then, also in this case, we can

conclude that $\mathcal{R}(\tau, x_0)/x_\tau$ on the side of $c(x - x_\tau) = 0$ opposite to the origin, and $\forall x \in \mathcal{R}(\tau, x_0)/x_\tau$, $(cx_\tau) c(x - x_\tau) > 0$. Hence, (16) holds.

Moreover, suppose now that $\psi(\bar{u}, x_0, t)$ enters \mathcal{D}_M with $\bar{u} = 0$, through $v^T x = 0$. Consider $\tau = \min\{t | (v^T \psi(0, x_0, \tau) = 0)\}$. The boundary of the set of points reachable in time $t \leq \tau$ is given by curves $\gamma_M(\alpha)$, $e^{A\alpha}x_0 + (I - e^{A\alpha})x_M$ and $e^{A\alpha}x_0 = \psi(0, x_0, \alpha)$ with $\alpha \in [0, \tau]$. Hence, any other control, which steers x_0 to the origin, cannot achieve a value of the cost function smaller than $|cx_\tau|$, since curve $\psi(0, x_0, \alpha)$ with $\alpha \in [0, \tau]$ can never be crossed for it is a piece of the boundary of the reachable set.

Control (13) is not an optimal solution to Problem 2 for, as discussed in [1], it does not provide finite time convergence to B_ρ for $x_0 \in \mathcal{D}_M$ s.t. $v^T x_0 = 0$. Rotating switching line $v^T x = 0$ by a $\theta > 0$, convergence to B_ρ within a finite time is achieved.

If $(cx_M)(cb) > 0$, \bar{u} as in (13), with $\theta \in (0, \cos^{-1}(|b^T v| \|b\|^{-1}))$, is again an optimal solution to Problem 2. In fact, since the center of \mathcal{C}_M belongs to $(R(\theta)v)^T x = 0$ with $\theta = \cos^{-1}(|b^T v| \|b\|^{-1})$ and $\psi(0, x_0, t)$ is convergent to the origin, for any θ in the interval above $x(t) = \psi(0, x_0, t)$, with $x_0 \in \mathcal{D}_M$ s.t. $(R(\theta)v)^T x_0 > 0$, reaches line $(R(\theta)v)^T x = 0$ in a point contained in \mathcal{D}_M . For $(cx_M)(cb) > 0$, $|\tilde{a}(\cdot)|$ monotonically decreases along $\psi(0, x_0, t)$ and hence \bar{u} is optimal. Conversely, if $(cx_M)(cb) \leq 0$, for any $x_0 \in \mathcal{D}_M$ between lines $cAx = 0$ and $v^T x = 0$, $|\tilde{a}(\cdot)|$ monotonically increases along $\psi(0, x_0, t)$. Let $\theta_{cA} = \cos^{-1}(|cAv| \|cA\|^{-1})$ denote the angle between lines $cAx = 0$ and $v^T x = 0$. For any $\theta \in (0, \theta_{cA})$, given $x_0 \in \mathcal{D}_M$ s.t. $v^T x_0 = 0$, $|\tilde{a}(\cdot)|$ is maximum at $\psi(0, x_0, \theta/\mu)$ on line $(R(\theta)v)^T x = 0$, and

$$\lim_{\theta \rightarrow 0^+} \{ \max_{t \in [0, T]} |c\psi(0, x_0, t)| \} = \lim_{\theta \rightarrow 0^+} |c\psi(0, x_0, \theta/\mu)| = |cx_0|.$$

Hence, given any $\theta_1 \in (0, \theta_{cA})$, one can always find $\theta \in (0, \theta_{cA})$ such that $|c\psi(0, x_0, \theta/\mu)| < |c\psi(0, x_0, \theta_1/\mu)|$. For any $x_0 \in \mathcal{D}_M$ which can be steered to $(R(\theta)v)^T x = 0$ with $u = 0$, an upper bound for $|c\psi(0, x_0, \theta/\mu)| - |cx_0|$ is given by $|c(e^{-A\frac{\theta}{\mu}} - I)v_\perp| \frac{M}{\mu} \|b\|$. Hence, for any $\epsilon > 0$, choosing $\theta \in (0, \theta_{cA})$ if $|c(e^{-A\frac{\theta_{cA}}{\mu}} - I)v_\perp| \frac{M}{\mu} \|b\| < \epsilon$, or $\theta \in (0, \theta_c)$ with θ_c solution to equation $|c(e^{-A\frac{\theta_c}{\mu}} - I)v_\perp| \frac{M}{\mu} \|b\| = \epsilon$, yields $\sup_{0 \leq t \leq T} |c\psi(\bar{u}, x_0, t)| - \tilde{A} < \epsilon$. Q.E.D.

4 Hybrid system control scheme

Control law (13) is clearly not feasible for the hybrid model \mathcal{M}_{4cyl} , introduced in Section 2, for three main reasons:

- (a) available torque is limited to 0 and to interval $[r_{min}M, M]$;
- (b) there is a delay between the time of injection and the time at which the corresponding torque is generated;
- (c) torque generation has to be synchronized with the powertrain dynamics.

Our solution to point (a) is to use bang-bang control everywhere in the state space, introducing an appropriate switching surface $\sigma(x) = 0$. To simplify discussion of point (b), set $x(k) := x(t_k)$ and $u(k) := u(t_k)$ for all k . Control signal $j_i(k)$, at current time t_k corresponding to an $E \rightarrow H$ transition of the i -th cylinder, will produce, at the next expansion run of the i -th cylinder, the torque $u(k+3)$. Such signal will feed the continuous system during the interval $[t_{k+3}, t_{k+4})$, steering the state from $x(k+3)$ to $x(k+4)$. A prediction of $x(k+3)$ is obtained from $x(k)$ by a forward integration of (5). Because of the synchronization constraint mentioned in (c), switchings of the torque u cannot occur, in general, exactly on $\sigma(x) = 0$. This issue is the main difficulty for devising a robust control strategy and is the main subject of this section.

If it is predicted that $x(t)$ will cross the switching surface $\sigma(x) = 0$ at some $t \in (t_{k+3}, t_{k+4})$, one can decide to switch j_i either at t_k or t_{k+1} . In certain conditions if switching is anticipated, an acceleration peak may be introduced. For this reason, we decide to switch always at time t_{k+1} , i.e.

$$\hat{j}_i(k) = \begin{cases} 0 & \text{if } x(k+3) \in B_\rho \\ \begin{cases} 0 & \text{if } \sigma(x(k+3)) \geq 0 \\ 1 & \text{if } \sigma(x(k+3)) < 0 \end{cases} & \text{if } x(k+3) \notin B_\rho \end{cases} \quad (17)$$

Control law (17) may fail to switch injection to 0 when state x enters B_ρ , and possibly go into a limit cycle. If $u(t) = M$, for $t \in [t_{k+3}, t_{k+4}]$, there may be crankshaft speeds such that delay $t_{k+4} - t_{k+3}$ causes the trajectory $\psi(M, x(t_{k+3}), t)$ to intersect B_ρ leaving both decision points $x(k+3)$ and $x(k+4)$ outside B_ρ . A necessary condition for this not to happen is to have $\|x(t_{k+4})\| < \|x(t_{k+3})\|$. The locus of points which under control $u = M$ increase their norm in time Δ is

$$\mathcal{N}_\Delta = \{x \mid \|x\| < \|e^{A\Delta}(x - x_M) + x_M\|\}. \quad (18)$$

Suppose there exist two distinct intersections of the boundaries of B_ρ and \mathcal{N}_Δ , namely $x_c^{(1)}$ and $x_c^{(2)}$, as shown in Figure 4. Let $\psi_{M-}(x, \alpha) = e^{-A\frac{\alpha}{\mu}}(x - x_M) + x_M$ be the point such that $\psi(M, \psi_{M-}(x, \alpha), \frac{\alpha}{\mu}) = x$. Let $\psi_c^{(1)}(\cdot)$ and $\psi_c^{(2)}(\cdot)$ be the curves $\psi_c^{(1)}(\alpha) = \psi_{M-}(x_A, \alpha)$, and $\psi_c^{(2)}(\alpha) = \psi_{M-}(x_B, \alpha)$, with $\alpha \in [-\frac{\pi}{2}, \pi]$, where x_A, x_B are such that $v^T x_A = 0$, $v^T x_B = 0$, $v_\perp^T(x_A - x_M) < 0$, $v_\perp^T(x_B - x_M) < 0$ and $x_c^{(1)} \in \psi_c^{(1)}(\cdot)$, $x_c^{(2)} \in \psi_c^{(2)}(\cdot)$. Let $x_D = \psi_c^{(2)}(\pi)$, and define the switching function in (17) as

$$\sigma(x) = \begin{cases} (v^T b / \|b\|)^{-1} v^T x & \text{if } (b_\perp^T x \leq b_\perp^T x_B) \vee (b_\perp^T x \geq b_\perp^T x_D) \\ (b / \|b\|)^T (x - x') & \text{if } b_\perp^T x_B < b_\perp^T x < b_\perp^T x_D \\ \text{with } x' \text{ s.t. } (b_\perp^T (x - x') = 0) \wedge (x' \in \psi_c^{(2)}) \end{cases} \quad (19)$$

Control $\hat{j}_i(k)$ in (17) switches on the curve described as follows (see Figure 4): the half line $v^T x = 0$ with $v_\perp^T(x - x_\rho) \leq 0$, and $x_\rho = -\rho v_\perp$; the arc of the boundary of B_ρ from x_ρ to $x_c^{(2)}$; the arc of $\psi_c^{(2)}(\cdot)$ from $x_c^{(2)}$ to x_D ; the half line $v^T x = 0$ with $v_\perp^T(x - x_D) \geq 0$. Such choice can prevent from the existence of limit cycles as illustrated in the following lemma.

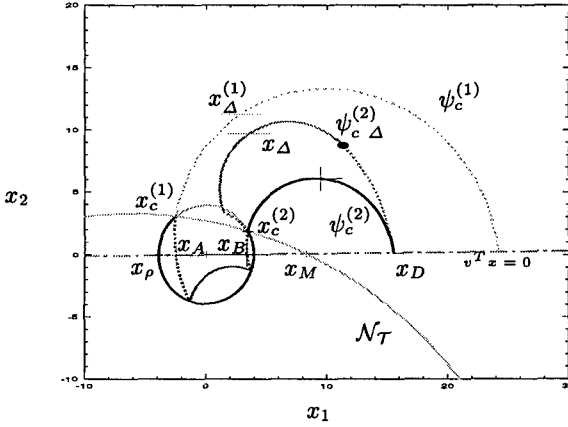


Fig. 4. Derivation of the hybrid switching surface.

Lemma 10. *If $\|x_M\| > \rho$, there exists a ω_{cmin} finite such that, substituting Δ in (18) for $\Delta_{max} = \frac{30}{\omega_{cmin}}$ (with ω_{cmin} in rpm), if a trajectory of system \mathcal{M}_{Acyl} under control law $\hat{j}_i(k)$ as in (17), with $\omega_c(t) \geq \omega_{cmin} = \forall t \geq 0$, intersects $\sigma(x) = 0$ through the arc $\psi_c^{(2)}(\alpha)$ with $\alpha \in [0, \pi]$ at some time $\bar{t} \in [t_{k+3}, t_{k+4}]$, then $x(t) \in B_\rho$ for all $t > t_{k+4} + \frac{3\pi}{2\mu}$.*

Proof. Note that $\lim_{\Delta \rightarrow 0} \mathcal{N}_\Delta = \mathcal{N}_0$, where \mathcal{N}_0 is the disk with boundary passing by both x_M and the origin 0 and center in $-\frac{M}{2\lambda}b$. From hypothesis $\|x_M\| > \rho$

$$\exists x_c^{(1)}, x_c^{(2)} \in \mathbb{R}^2, \text{ with } x_c^{(1)} \neq x_c^{(2)}. \quad (20)$$

By continuity and monotonicity, there exists a finite time $\Delta_{max1} > 0$ and correspondingly a speed $\omega_{cmin1} = \frac{30}{\Delta_{max1}}$, such that for all $\Delta < \Delta_{max1}$, (20) holds and (19) is well-defined.

Under the hypotheses of the Lemma, all points $x(k+4)$ belong to the set

$$S_\Delta = \left\{ x : x = e^{A\delta} \psi_c^{(2)}(\alpha), \text{ with } \alpha \in [0, \pi] \text{ and } \delta \in [0, \Delta] \right\}, \quad (21)$$

with boundaries: $\psi_c^{(2)}(\alpha)$ with $\alpha \in [0, \pi]$; $\psi_{c\Delta}^{(2)}(\cdot)$, where $\psi_{c\Delta}^{(2)}(\alpha) = e^{A\Delta} \psi_c^{(2)}(\alpha)$, with $\alpha \in [0, \pi]$; $\psi(0, x_D, \delta)$, with $\delta \in [0, \Delta]$; $\psi(0, x_B, \delta)$, with $\delta \in [0, \Delta]$.

If $x(k+4) \in B_\rho$, $\hat{j}_i(k+1) = 0$ and $x(t) \in B_\rho \forall t \geq t_{k+4}$. If $x(k+4) \notin B_\rho$, $\hat{j}_i(k+1) = 1$. If $S_\Delta \cap \psi_c^{(1)}(\cdot) = \emptyset$, then the trajectory $\psi(M, x(k+4), t)$ originating at $x(k+4)$ under $u(t) = M$ enters B_ρ through the arc from $x_c^{(1)}$ to $x_c^{(2)}$. Since $\|x_M\| > \rho$, $\psi(M, x(k+4), t)$ reaches B_ρ at time $t_\rho \leq t_{k+4} + \frac{3\pi}{2\mu}$.

By construction there exists a $\bar{K} > k+4$ such that $x(\bar{K}-1) \notin B_\rho$ and $x(\bar{K}) \in B_\rho$, since $x(\bar{K}-1) \notin \mathcal{N}_\Delta$, with $t_{\bar{K}-1} < t_\rho \leq t_{\bar{K}}$. Consequently injection switches to zero, and $x(t) \in B_\rho, \forall t \geq t_\rho$. $\lim_{\Delta \rightarrow 0} \psi_{c\Delta}^{(2)}(\cdot) = \lim_{\Delta \rightarrow 0} S_\Delta = \psi_c^{(2)}(\cdot)$

Necessarily $\mathcal{S}_\Delta \cap \psi_c^{(1)}(\cdot) = \emptyset$. By continuity and monotonicity, there exists a finite time $\Delta_{max} > 0$ with $\Delta_{max} < \Delta_{max1}$ and correspondingly a speed $\omega_{cmin} = \frac{30}{\Delta_{max}}$ such that for all $\Delta < \Delta_{max}$ $\mathcal{S}_\Delta \cap \psi_c^{(1)}(\cdot) = \emptyset$. Q.E.D.

To find Δ_{max} of Lemma 10, we derive a function $f(\Delta)$ which is positive if $\mathcal{S}_\Delta \cap \psi_c^{(1)} = \emptyset$, negative otherwise. We first determine the trajectory $\psi_c^{(\Delta)}$, where $\psi_c^{(\Delta)}(\alpha) = \psi_{M-}(\bar{x}_\Delta, \alpha)$, with $\alpha \in [0, \pi]$, and $v^T \bar{x}_\Delta = 0$ such that $\psi_c^{(\Delta)}$ is tangent to $\psi_{c\Delta}^{(2)}$ at point $x_\Delta = \psi_{c\Delta}^{(2)}(\alpha_\Delta)$. At point x_Δ

$$-\left. \frac{\partial \psi_{c\Delta}^{(2)}(\alpha)}{\partial \alpha} \right|_{\alpha=\alpha_\Delta} // A \left(\psi_{c\Delta}^{(2)}(\alpha_\Delta) - x_M \right). \quad (22)$$

Condition (22) is satisfied for $\alpha_\Delta = \mu\Delta + \cos^{-1} v_\perp^T \vec{v}ers(e^{A\Delta} x_M - x_M)$. Consider $x_\Delta^{(1)} = \psi_c^{(1)}(\alpha_\Delta^{(1)})$, with $\alpha_\Delta^{(1)} = \cos^{-1} v_\perp^T \vec{v}ers(x_\Delta)$, and define $f(\Delta) = \|x_\Delta^{(1)} - x_M\| - \|x_\Delta - x_M\|$. Control \hat{j}_i does not lead to limit cycles when crossing $\psi_{c\Delta}^{(2)}$ if

$$f(\Delta) = \|x_\Delta^{(1)} - x_M\| - \|x_\Delta - x_M\| > 0. \quad (23)$$

Convergence of the trajectories intersecting the switching surface on $\psi_c^{(2)}$ is addressed in the previous lemma. However, we need to prove convergence for all initial conditions.

Let us compare the trajectories when intersecting the line $v^T x = 0$, which is the optimal switching surface for the continuous case. In the hybrid case, the switching occurs with some delay due to the discretization of the decision points. There are two cases to analyze: a) $v_\perp^T x < 0$, i.e. the intersection is to the left of B_ρ , b) $v_\perp^T (x - x_M) > 0$, i.e. the intersection is to the right of B_ρ .

In both cases, we can bound the difference between the value of the norm of the points on the optimal trajectory and those corresponding to the hybrid control law. We can also compare the performance of the hybrid control law with respect to accelerations peaks. Note that since the switchings occur at different points, we compare the points on the next intersection on the optimal switching line $v^T x = 0$, to be able to compare norms.

Let x_{M0} be such that $v^T x_{M0} = 0$, with $v_\perp^T x_{M0} < 0$. Consider the point $x_{M0}^{(2)}$ reached under control $u = M$ for Δ seconds, i.e., the point that corresponds to the switching of control occurring as far as possible from the optimal switching surface. The hybrid trajectory then reaches $v^T x = 0$ at point $x_{M0}^{(3)}$ under control $u = 0$, that is

$$x_{M0}^{(2)} = e^{A\Delta}(x - x_M) + x_M, \quad x_{M0}^{(3)} = \left(e^{A\frac{1}{\mu} \cos^{-1} v_\perp^T \vec{v}ers(x_{M0}^{(2)})} \right) x_{M0}^{(2)}.$$

Let $x_{M0}^{(1)} = e^{A\frac{\pi}{\mu}} x_{M0}$ be the point reached by the continuous trajectory from x_{M0} , we define

$$w^0(x_{M0}, \Delta) = \|x_{M0}^{(3)}\| - \|x_{M0}^{(1)}\|, \quad w_a^0(x_{M0}, \Delta) = |cy_\perp| e^{-\frac{\lambda}{\mu} \alpha_c} w^0(x_{M0}, \Delta),$$

the increase in norm due to non optimal switching and the corresponding increase in the peak following the transition, where $\alpha_c = \cos^{-1}(v^T y)$.

Let x_{0M} be such that $v^T x_{0M} = 0$ and $v_\perp^T(x_{0M} - x_M) > 0$, and analogously to the previous case, consider the points $x_{0M}^{(1)} = \psi(M, x_{0M}, \frac{\pi}{\mu})$ and $x_{0M}^{(2)} = \psi(0, x_{0M}, \Delta)$ respectively on the continuous and on the hybrid trajectory, the latter subsequently reaches $x_{0M}^{(3)} = \psi(M, x_{0M}^{(2)}, \frac{1}{\mu} \cos^{-1} \frac{v_\perp^T x_{0M}^{(2)}}{\|x_{0M}^{(2)}\|})$, and with the same meaning we define

$$w^M(x_{0M}, \Delta) = \|x_{0M}^{(3)}\| - \|x_{0M}^{(1)}\|, \quad w_a^M(x_{0M}, \Delta) = |cy_\perp| e^{-\frac{\lambda}{\mu} \alpha_c} w^M(x_{0M}, \Delta).$$

The two functions $w^0(x_{M0}, \Delta)$ and $w^M(x_{0M}, \Delta)$ bound the increase in norm due to the late switching for the $u = M \rightarrow u = 0$ transition and for the $u = 0 \rightarrow u = M$ transition of torque value respectively. Not surprisingly, both functions are monotonically decreasing with respect to $\|x_{M0}\|, \|x_{0M}\|$ and $\omega_c = \frac{30}{\Delta}$, that is the increase in norm is less for high revolution speeds and for initial conditions that are far from B_ρ .

Now we are ready to discuss the convergence of the control law. Here we have to show that the trajectories originating from points away from B_ρ tend towards B_ρ . To do so, we show that the norm of two consecutive intersection points of the same trajectory under the hybrid control law on $v^T x = 0$, $v_\perp^T(x - x_D) \geq 0$ decreases. Because of the monotonic behavior of $w^0(x_{M0}, \Delta)$ and $w^M(x_{0M}, \Delta)$, the worst initial condition is the closest to the origin which may not switch between $\psi_c^{(1)}$ and $\psi_c^{(2)}$. To identify such initial condition, consider figure 5.

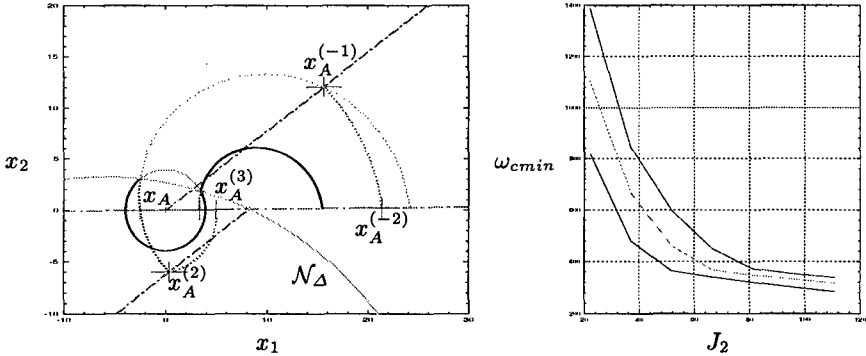


Fig. 5. Limiting trajectory and Parameters ranges.

The trajectory obtained by back integration from x_A with torque $u = M$ switches to $u = 0$ at some point x_s between $v^T x = 0$ and $\hat{v}^T x = 0$, where $\hat{v} = e^{-\lambda \Delta} e^{A \Delta} v$. The worst point is $x_s = x_A^{(-1)} = \psi_c^{(1)}(\alpha_\Delta^{(-1)})$, with $\alpha_\Delta^{(-1)} \in [0, \pi]$

solution to $\hat{v}^T \psi_c^{(1)}(\alpha) = 0$. In fact points on $\hat{v}^T x = 0$ correspond to a switching occurring with the maximum delay Δ with respect to the ideal switching. Now, let $x_A^{(-2)} = e^{-A\Delta} x_A^{(-1)}$ be the point on $v^T x = 0$ from which the latest switching from $u = 0$ to $u = M$ leads to x_A . All other points on $v^T x = 0$ with $0 < v_\perp^T x < v_\perp^T x_A^{(-2)}$ fall into B_ρ and are characterized by Lemma 10. Let $x_A^{(3)}$ be the point on $v^T x = 0$ after a late switching from x_A , then $\|x_A^{(-2)}\| - \|x_A^{(3)}\|$ bounds from below the decrease in norm on the trajectory between two consecutive intersection points with $v^T x = 0$, $v_\perp^T(x - x_A^{(-2)}) \geq 0$, due to the way we picked the switching surfaces and x_A . If indeed

$$\|x_A^{(-2)}\| - \|x_A^{(3)}\| = \|x_A^{(-1)}\| e^{-\lambda\Delta} - \|x_A\| e^{\frac{\lambda}{\mu}\pi} - w^0(x_A, \Delta) > 0 \quad (24)$$

then, all the other initial conditions farther than $x_A^{(-2)}$ from the origin yield trajectories with decreasing norms, and the control law is convergent.

In the end, control law \hat{j}_i is convergent for all $\omega_c \geq \frac{30}{\Delta}$, if Δ is such that

- C1** $\partial\mathcal{N}_\Delta \wedge \partial B_\rho \neq \emptyset$, i.e. if $\exists x_c^{(1)}, x_c^{(2)} \in \mathbb{R}^2$ so to construct $\sigma(x)$ as in (19),
- C2** limit cycle condition (23) holds,
- C3** norm convergence condition (24) holds.

Performance of the hybrid control law with respect to acceleration peaks can be compared to the continuous one.

Proposition 11. *For any choiche of the parameters, such that conditions C1, C2, C3 are satisfied, and $(cx_M)(cb) > 0$, there exists a $\rho_A > 0$ such that:*

Given x_0 with $\|x_0\| > \rho_A e^{-\frac{\pi\lambda}{\mu}}$, let $\bar{x} = \psi(\bar{u}, x_0, \bar{t})$, with $\bar{t} \in [0, T]$ and \bar{u} as in (13), be such that $\max_{t \in [0, T]} |c\psi(\bar{u}(x), x_0, t)| = |\bar{c}\bar{x}|$. Let \hat{T} be such that $\psi(\hat{u}, x_0, \hat{T}) \in \partial B_\rho$, with \hat{u} given by \mathcal{M}_{4cyl} under control $\hat{j}_i(k)$. Let $\hat{x} = \psi(\hat{u}, x_0, \hat{t})$, with $\hat{t} \in [0, \hat{T}]$, be such that $\max_{t \in [0, \hat{T}]} |c\psi(\hat{u}, x_0, t)| = |\hat{c}\hat{x}|$, and let $x^v = \psi(\hat{u}, x_0, t^v)$, with $t^v \in [0, \pi/\mu]$ be such that $v^T x^v = 0$. Then, if $v^T x_0 < 0$

$$0 \leq |\hat{c}\hat{x}| - |\bar{c}\bar{x}| \leq w_a^0(x^v, \Delta_{max}) \leq w_a^0(-\rho_A v_\perp, \Delta_{max}), \quad (25)$$

where $\Delta_{max} = \frac{30}{\omega_{cmin}}$. If $v^T x_0 \geq 0$

$$0 \leq |\hat{c}\hat{x}| - |\bar{c}\bar{x}| \leq w_a^M(x^v, \Delta_{max}) \leq w_a^M(\rho_A v_\perp, \Delta_{max}). \quad (26)$$

In particular $\hat{j}_i(k)$ is optimal for Problem 1, i.e. $|\hat{c}\hat{x}| - |\bar{c}\bar{x}| = 0$ if x_0 belongs to either $\mathcal{R}_3 = \{x \mid v^T x < 0, y^T(x - x_M) < 0\}$ or $\mathcal{R}_4 = \{x \mid v^T x > 0, y^T x > 0\}$.

5 Experimental Results

The proposed cut-off control strategy has been implemented and tested at Magneti-Marelli Engine Control Division on a commercial car, a 16 valve 1400 cc engine car equipped with drive-by-wire electronics. The engine control electronics is a 4LV Magneti Marelli on board computer based on a 25MHz 32-bit

Althair Motorola microprocessor with fixed point arithmetic unit. The experiment was carried out driving the car in the test ring and measuring the important parameters and variables that determine the performance of the control strategy.

5.1 Model Parameters

The continuous powertrain dynamics of the car is described by

$$A^p = \begin{bmatrix} 0 & 1 & -7.556 \\ -448.1 & -5.186 & 30.87 \\ 3.042 & .02773 & -.2105 \end{bmatrix}, \quad b^p = \begin{bmatrix} 0 \\ 15.05 \\ 0 \end{bmatrix},$$

and have dominant pole $\lambda_1 = -0.05460$ and complex poles $\lambda \pm j\mu = -2.671 \pm j21.54$, where the parameters of the model were carefully identified on the actual engine. The output map from states $\zeta = [\alpha_e, \omega_c, \omega_p]^T$ to the vehicle acceleration a (in meters per square second) is $c^p = [.8022, .007313, -.05525]$. The maximum torque achievable when the engine is idle with spark advance $r = 1$, is $M = 12.41$ Nm. The state space transformation matrix

$$P = \begin{bmatrix} 0.05155 & 0.04924 & 7.252 \\ 20.83 & 0.12780 & -0.5979 \\ 0.1596 & -0.9517 & 7.195 \end{bmatrix}$$

allows us to rewrite the powertrain dynamics in the uncoupled form (4) :

$$\begin{bmatrix} b' \\ b \end{bmatrix} = \begin{bmatrix} .7410 \\ 1.923 \\ -14.32 \end{bmatrix}, \quad [c', c] = [-1.857 \cdot 10^{-3}, 3.726 \cdot 10^{-2}, -2.523 \cdot 10^{-3}].$$

The acceleration perception threshold was set to 0.03 the acceleration due to gravity. Hence ρ is 7.872.

5.2 Convergence Analysis

Before starting the experimentation of the proposed control algorithm, a corner analysis with respect to the vehicle inertia momentum J_2 was carried out, in order to establish the minimum crankshaft speed ω_{cmin} for which (17) converges. Figure 5 reports the minimum crankshaft speed, obtained from conditions **C1**, **C2**, **C3**, for a given J_2 and for spark advance modulation factor r_i in $\{0.6, 0.8, 1\}$. It is interesting to note that, acting on the spark advance, one can reduce substantially ω_{cmin} . For the identified value $J_2 = 73.95$, we have $\omega_{cmin} \simeq 396rpm$, $356rpm$, $329rpm$, respectively for $r_i = 1, 0.8, 0.6$.

To prevent engine from stopping, cut-off strategies are usually not applied for $\omega_c < \simeq 1000rpm$. With the identified parameters, for $\omega_c = 1000rpm$, $f(\Delta) \simeq 8.22$, $\|x_A^{(-2)}\| - \|x_A^{(3)}\| \simeq 7.92$, hence convergence of (17) is ensured.

5.3 Discussion

The cut-off control strategy is applied after the accelerator pedal is released and the manifold pressure is equal to 300 mbar. A simple Leunberger observer is used to estimate the state of the powertrain's continuous dynamics, from which the oscillatory components \hat{x} are derived. The initialization procedure computes also the observer gain matrix L , to obtain satisfactory convergence of the observer. The switching function $\sigma(x)$, given in section 4, has been described by piecewise linear approximation with 20 points.

In the sequel the performances achieved by the proposed cut-off strategy are compared with the performance of a currently implemented open-loop strategy and with an instantaneous uncontrolled cut-off operation. Figure 6 shows the evolution of \hat{x} components of observer state during a cut-off operation. The switching curve $\sigma(x) = 0$, as it is approximated in the implementation, is reported along with B_p . Next, the evolution of the crankshaft speed, in rpm, is reported next. Only a significant positive peak is recognizable in the crankshaft speed. Such behavior is also shown in the next plot in terms of vehicle acceleration. As expected, once injection is set to zero permanently $\tilde{a}(t)$ remains bounded within the perception threshold. Finally, the last plot shows the evolution of the injection signal.

The experimental results fully confirm the theory of Section 4. When comparing this strategy with the open loop strategy used in the previous implementation, we observed a much better performance in terms of acceleration peaks and driving comfort. One could argue that this improvement, predicted by the theory, has been obtained with a more expensive implementation. Actually our strategy resulted in a much smaller memory occupation and in a very reasonable CPU time. In fact, the code size was 70% and the data size was 50% of the ones needed by the previously implemented strategy. The CPU power needed to run the control algorithm was about 1% of the available computing power. In addition, our strategy, being closed loop, needed much less tuning effort than the open loop strategy.

6 Conclusions and future work

In this paper, we presented a novel approach to engine control in the cut-off region, based on a hybrid model of the torque generation and of the power-train dynamics in a four-stroke engine. A control problem on this hybrid system is defined and solved using a sequence of approximations. The properties of the control law so obtained have been characterized, thus offering better confidence on the quality of the results with respect to commonly used heuristic, open loop, approaches. In addition, since the control law is closed loop, expensive tuning processes can be avoided yielding a commercially appealing solution. We expect to see the final version of the control laws in products by the first half of 1998. In addition, we expect to extend the approach to the problem of idle speed control that shares several key features with the cut-off control problem.

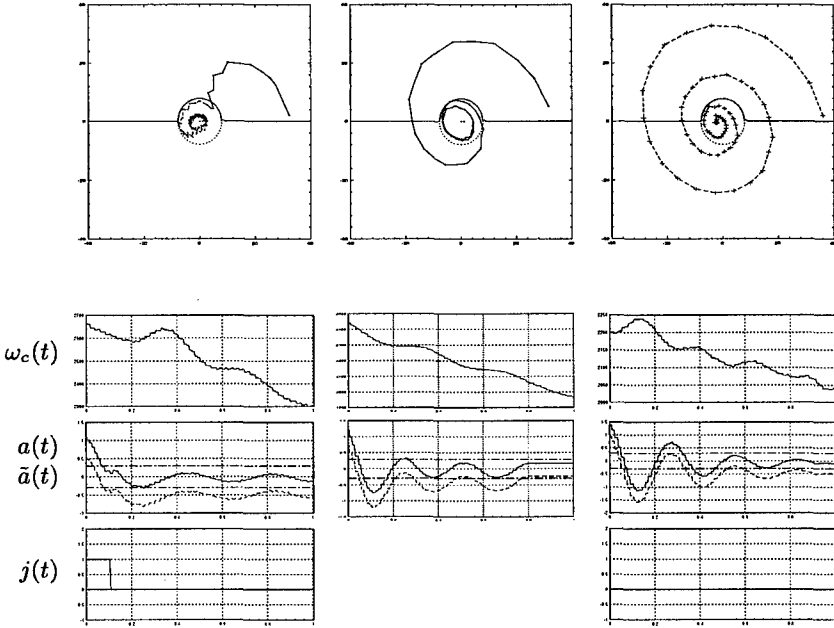


Fig. 6. Evolutions in the \hat{x} sub-space of observer state during cut-off operations. From left to right: the proposed control, a currently implemented strategy and instantaneous cut-off. State samples are linked by a continuous arc if in the previous cycle injection took place, dotted one if it did not. Evolution of the corresponding revolution speed (in rpm), vehicle acceleration (in $m^2/second$, dotted line represents total acceleration $a(t)$ and continuous line represents the oscillating component $\tilde{a}(t)$), and injection signal.

7 Acknowledgments

This research has been partially sponsored by PARADES, a Cadence, Magneti-Marelli and SGS-Thomson GEIE, and by CNR. We wish to acknowledge the support of the management of Magneti Marelli, and, in particular of Drs. Pecchini, Mortara and Romeo, without which this work would not have been possible. ISI provided the X-Math environment to carry out the simulation and the development of the control law. Dr. Alberto Ferrari developed two powerful simulation environments, both including X-Math. One is based on Ptolemy, a University of California simulation and design tool for heterogeneous systems, the other on BoNES, a commercial simulation tool from the Alta group of Cadence. These environments have been essential to develop, tune and implement the control strategy. Dr. Luca Benvenuti offered insightful comments on the results of the paper.

References

1. A. BALLUCHI, M. D. BENEDETTO, C. PINELLO, C. ROSSI, AND A. SANGIOVANNI-VINCENTELLI, *Cut-off in engine control: a hybrid system approach*, in 36th CDC, San Diego, CA, 1997.
2. J. B. HEYWOOD, *Internal Combustion Engine Fundamentals*, McGraw-Hill Book Co., Inc., NY, 1989.
3. C. HORN AND P. J. RAMADGE, *Robustness issues for hybrid systems*, in 34th CDC., New Orleans, LA, 1995, pp. 1467-1472.
4. L. HOU, A. MICHEL, AND H. YE, *Stability analysis of switched systems*, in 35th CDC, Kobe, Japan, 1996, pp. 1208-1212.
5. A. S. MORSE, ed., *Control using logic-based switching*, vol. 222 of Lecture notes in control and information sciences, Springer-Verlag, London, U.K., 1997.
6. T. NIINOMI, B. H. KROGH, AND J. E. R. CURY, *Synthesis of supervisory controllers for hybrid systems based on approximating automata*, in 34th CDC., New Orleans, LA, 1995, pp. 1461-1466.
7. S. PETTERSSON AND B. LENNARTSON, *Stability and robustness for hybrid systems*, in 35th CDC, Kobe, Japan, 1996, pp. 1202-1207.
8. A. SANGIOVANNI-VINCENTELLI, *Embedded system design and hybrid systems*, in Control using logic-based switching, A. S. Morse, ed., vol. 222 of Lecture notes in control and information sciences, Springer-Verlag, London, U.K., 1997, pp. 17-38.
9. H. J. SUSSMANN, *Lie brackets, real analyticity and geometric control*, in Differential Geometric Control Theory, R. W. Brockett, R. S. Millman, and H. J. Sussmann, eds., vol. 27 of Progress in Mathematics, Birkhäuser, Boston Basel Stuttgart, 1983, pp. 1-117.
10. H. YE, A. N. MICHEL, AND L. HOU, *Stability theory for hybrid dynamical systems*, in 34th CDC., New Orleans, LA, 1995, pp. 2679-2684.

Hybrid Control of Automotive Powertrain Systems: A Case Study *

Ali Beydoun,** Le Yi Wang***, Jing Sun and Shiva Sivashankar [†]

1 Introduction

This paper is concerned with the problem of hybrid control strategies for automotive powertrain systems. Automotive systems represent an important class of practical hybrid systems which are characterized by the following features:

1. The systems are inherently hybrid, i.e., hybrid control is not merely a choice. This is exemplified by transmission gear positions (discrete) and engine throttle control (analog).
2. System dynamics are highly nonlinear and contain parametric errors and structural uncertainties. Any model that is suitable for control system development will inevitably contain large modeling errors.

In this paper, a hybrid control design approach is used to develop control strategies for coordination of automotive engine and transmission systems. The main goal is to improve system performance, fuel economy, robustness, and other performance specifications. The design procedure follows closely the main ideas of the method introduced recently in [4]. The method employs performance indices in guiding both analog and discrete control actions such that robust stability and performance of closed-loop hybrid systems are maintained, in the presence of modeling errors, disturbances, and structural uncertainties. The method, however, must be modified significantly in this case study to accommodate practical constraints, including actuator saturations, gear shifting limitations, and real-time computation requirements.

1.1 Automotive Powertrain Hybrid Systems

Consider a typical automotive powertrain system shown in Figure 1. Depending on engine mechanical configurations, control signals may include throttle angle θ

* This research is supported in part by NSF Grant ECS-9634375 and several research grants from Ford Motor Company.

** Advanced Vehicle Technology, Ford Motor Company, Mail Drop 76, 21500 Oakwood Blvd., Dearborn, Michigan 48121.

*** Department of Electrical Engineering, Wayne State University, Detroit, Michigan 48202, Tel.: 313-577-4715, Fax: 313-577-1101, Email: lywang@ece.eng.wayne.edu

[†] Ford Motor Company, Scientific Research Laboratory, P.O. Box 2053, MD 2036, Dearborn, Michigan 48121-2053

(which is controlled either manually by the driver's foot pedal, or electronically by engine controllers), spark advance δ , exhaust gas recirculation *EGR*, air/fuel ratio *AFR*, transmission gear position *G*, swirl control valve (*SCV*), etc. The outputs of the system may include vehicle speed, emission pollutants (such as *HC*, *CO*, and *NO_x*), fuel consumption, etc. There are certain discrete actions which cannot be controlled by powertrain control systems. For instance, the action of the driver to switch on cruise control is an uncontrollable discrete action. On the other hand, gear positions and *SCV* actions are discrete control variables.

The design objectives include fast and smooth acceleration responses to the driver's pedal commands; low fuel consumption; low levels of tailpipe emissions; and reduction of noise and vibration; among others. These performance measures are to be achieved in a wide range of operating conditions in the presence of discrete uncertainties such as cruise control switching, and analog disturbances such as load changes from road conditions.

Several important characteristics of powertrain systems have rendered classical control design methodologies ineffective in developing high efficient modern powertrain control systems. First, Powertrain systems are inherently hybrid. Gear selections and switching of cruise control are clearly discrete actions which can only assume a finite number of values. In contrast, θ , δ , *EGR*, *AFR* are analog signals. Discrete actions such as gear shifts will cause a rapid change of internal states (such as engine speed) and system dynamics.

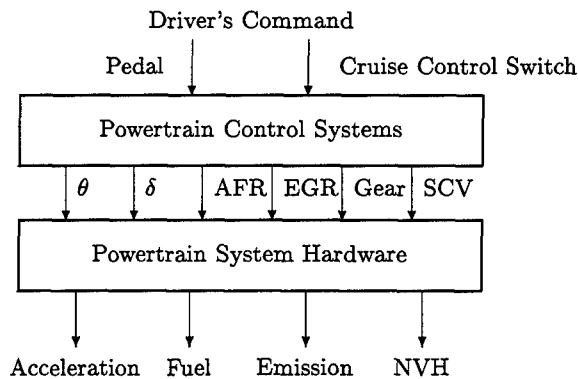


Fig. 1. Powertrain Systems

Second, powertrain hybrid systems are highly nonlinear, operate in a wide range of conditions and have to tolerate large disturbances and uncertainties, have both controllable and uncontrollable discrete events. To achieve a total

trade-off among competing performance specifications, it becomes necessary to apply complicated control mechanism and search for optimal combinations of control variables.

Besides the nonlinear and hybrid nature of powertrain systems, hardware limitations also impose certain constraints on control strategies. First, actuators such as throttle and spark have significant saturation limits, determined either by mechanical design limits or operating conditions. Second, the control decisions must be made with limited on-board computation resources. Third, frequent gear shifting and gear hunting should be avoided.

1.2 Generic Hybrid Control Structures

Automotive powertrain systems are special cases of the generic state-space hybrid systems depicted in Figure 2.

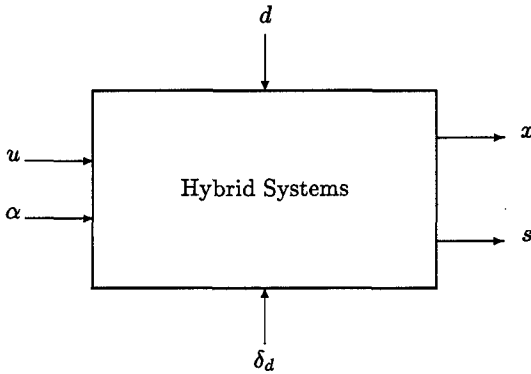


Fig. 2. Hybrid Systems

Mathematically, hybrid systems in state-space form are expressed as

$$\begin{aligned}\dot{x} &= f(x, u, d; s; t) + \Delta(x, u, d; s; t), \\ s &= DES(\delta_d, \alpha)\end{aligned}\tag{1}$$

where $x(t) \in \mathbb{R}^n$ is the analog state, $u(t) \in \mathbb{R}^m$ the analog control input, $d \in \Omega_d$ the analog disturbances; $s \in \mathbb{Q}$ is the discrete state, $\delta_d \in \Omega_\delta$ is the uncontrolled discrete action. Ω_δ is called *discrete uncertainty set*. $\alpha(t) \in \Sigma$ is the discrete control action, and Σ is a finite set of admissible discrete control actions. $\Delta \in \Omega$ represents model uncertainties. Ω is called *model uncertainty set*. f is assumed to be continuous in x and u . DES is a discrete event automaton.

The discrete action σ (i.e., δ_d or α) takes the form of $\sigma = (s, s_1)$ where s is the old discrete state and s_1 the new one. Associated with a discrete action at t , the system nominal dynamics will switch from $\dot{x} = f(x, u, d; s; t)$ to $\dot{x} = f(x, u, d; s_1; t)$, and the state x will jump from $x(t_-)$ to $x(t_+)$.

For the powertrain systems in this study, the discrete-event system, which includes the gear set and its shift schedule, contains essentially the four gear positions of gears 1 through 4. For the purpose of control design, the powertrain systems will be modeled as a second-order nonlinear system without time-delay. This approximation, combined with engine-to-engine deviations, environmental changes, inherent time-delays, external disturbances and structural uncertainty, results in large modeling errors. As a result, control strategies must not only deliver satisfactory performance under a nominal condition, but also guarantee robustness against all such uncertainties.

2 Simplified Powertrain Models

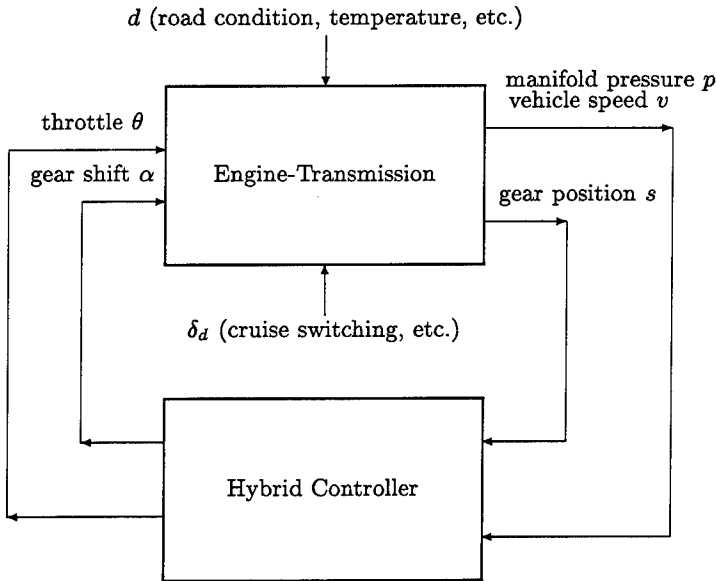


Fig. 3. Engine-Transmission Hybrid Control Systems

For this case study, we employ a simplified powertrain system model which contains an engine, a static transmission system and a simple buffer representing a torque convertor. Its structure is depicted, together with a hybrid controller, in Figure 3. The engine model is based on a 1.8L 4 cylinder port fuel injection engine. The model was developed by Ken Butts from Ford Research Lab in collaboration with Mathworks Inc. [2], based on a paper by P.R. Crossley and J.A. Cook [1]. This model can be obtained free of charge from Mathworks. While the model does not quantitatively represent a modern production powertrain system, it is significantly representative in its structure and essential features. As a result, the control strategies developed based on this model can be readily extended to cover more realistic powertrain systems.

2.1 Models

We assume that the powertrain system is equipped with an electronic throttle. As a result, the throttle angle or air charge becomes the primary analog control variable. The air charge rate \dot{m} (g/s) through the throttle body is a function of the throttle angle θ (degree) and manifold pressure p (KPa),

$$\dot{m} = f(\theta)g(p),$$

where

$$f(\theta) = 2.821 - 0.05231\theta + 0.10299\theta^2 - 0.00063\theta^3$$

and

$$g(p) := \begin{cases} \frac{2}{p_a} \sqrt{pp_a - p^2}, & p > 0.5p_a; \\ 1, & p \leq 0.5p_a. \end{cases}$$

In a normal operating condition, the atmospheric pressure p_a is approximately $p_a = 100$ KPa.

The manifold dynamics is usually modeled as a first order linear system which relates air charges to the changes in manifold pressure p :

$$\dot{p} = k(\dot{m} - \dot{M})$$

where $k = 0.5786$ is a constant under some idealized assumptions, \dot{M} (g/s) is the air charge rate into the cylinders. The induction of air into engine cylinders can be modeled as a nonlinear function of the engine speed N (rad/s) and manifold pressure p ,

$$\dot{M} = -0.366 + 8.979pN_r - 337N_r p^2 + 0.01pN_r^2$$

where N_r is the engine rotational speed in rad/sec, which is related to the engine speed N in RPM by

$$N_r = \frac{2\pi}{60}N.$$

The engine speed is related to the vehicle speed by a transmission system which defines the ratio of engine speed to vehicle speed for each gear position. We assume that the transmission has 4 gear positions with the corresponding ratios

$$\frac{N}{v} = \beta(i), \quad i = 1, 2, 3, 4$$

with $\beta(1) = 28.10, \beta(2) = 15.69, \beta(3) = 10.30, \beta(4) = 7.26$, in (rad/s)/(m/s). To capture the damping effect of torque convertors, the transient values of β when gear shifting occurs will be an exponential function from the old value to the new value, taking an average of 0.5 second to finish the transition.

Combining the above equations, we obtain the first dynamic equation

$$\begin{aligned} \dot{p} &= k\dot{m} - k(-0.366 + 8.979pN_r - 337N_r p^2 + 0.01pN_r^2) \\ &= a_0 + a_1 p \beta(i) v + a_2 \beta(i) v p^2 + a_3 p (\beta(i))^2 v^2 + u \end{aligned}$$

where $a_0 = -k(-0.366)$, $a_1 = -kk_n(8.979)$, $a_2 = -kk_n(-337)$, $a_3 = -kk_n^2(0.01)$, and $k_n = \frac{2\pi}{60}$; and for control design purposes, we define

$$u = k\dot{m}$$

as the control signal. If the throttle is not saturated, the real control variable, i.e., the throttle angle, can be determined by the throttle body mapping for a given u .

For a 4-cylinder 4-stroke engine, it is easy to compute that the air charge for each individual cylinder per intake stroke is

$$M_c = \dot{M} \frac{60}{2N}.$$

Now, the brake torque T_e (Nm) produced by the engine can be experimentally established as a nonlinear regressed function of cylinder air charge M_c , spark advance δ (degree), engine speed N_r (rad/s), exhaust gas recirculation EGR , and air-to-fuel ratio AFR . In this model, EGR is fixed as a constant, hence does not appear in the model.

$$\begin{aligned} T_e &= 2 \times 0.7376(-181.3 + 379.36M_c + 21.91AFR - 0.85AFR^2 + 0.26\delta \\ &\quad - 0.028\delta^2 + 0.027N_r - 0.000107N_r^2 + 0.00048N_r\delta + 2.55\delta M_c - 0.05\delta^2 M_c). \end{aligned}$$

To maximize engine efficiency, it is usually desirable that the spark advance is selected such that the engine torque output is maximized. Such a value of the spark is called MBT (maximum brake torque) spark. In this model, the MBT spark can be computed from the nominal model as a function of the air charge and engine speed

$$\delta = \frac{0.26 + 0.00048N_r + 2.55M_c}{2(0.0028 + 0.05M_c)}.$$

Furthermore, to maximize the efficiency of the three-way catalyst, the air-to-fuel ratio must be close to the stoichiometric value which in our case is 14.6.

The vehicle acceleration follows the Newton's law

$$\dot{v} = \frac{F}{W}$$

where F is the net force on the vehicle for acceleration, W the vehicle weight in Kg, and g is the gravity acceleration in (m/s^2) . For the (hypothetical) vehicle we are considering, $g = 9.8$, $W = 1134$ Kg (or 2500 lbs). F is related to engine power and load by

$$F = \frac{P_e - P_L}{v}$$

where P_e and P_L are engine brake power and load (Watts), respectively. The engine brake power can be computed as $P_e = \frac{\pi}{30}NT_e$, where N is the engine speed in RPM and T_e is the engine brake torque in Nm.

A typical expression for the load is $P_L = a + bv^2$, where the values of a and b depend on vehicles and road conditions. An example of these values are $a = 2.011 \times 10^{-3}$ and $b = 1.5 \times 10^{-6}$. For hilly roads, a and b become bigger.

In summary, the system we are considering can be modeled as a second-order nonlinear system,

$$\begin{aligned}\dot{v} &= f_v(p, v, i) \\ \dot{p} &= f_p(p, v, i) + u\end{aligned}$$

where

$$f_v(p, v, i) = \frac{1}{W} \frac{2\pi T_e N - (a + bv^2)}{v}$$

$$f_p(p, v, i) = a_0 + a_1 k_n p \beta(i) v + a_2 \beta(i) v p^2 + a_3 p \beta^2(i) v^2.$$

For design purposes, the transport delay has been omitted. The robust design we shall employ can be shown to guarantee robustness against modeling errors and delays. These will be demonstrated in our simulation.

2.2 Control Configurations

The hybrid control system in Figure 3 will consist of:

1. A robust analog controller which feeds back from the analog states (vehicle speed and manifold pressure) and discrete state (gear position) to determine the throttle position so that tracking control can be achieved in the presence of modeling errors and external disturbances.
2. A discrete event model which, other than initial states, has four states representing gear positions. Some constraints will disable certain transitions

between the states. Most obviously, the upshift gear skip from 1 to 4 is not allowed. Other gear skips are allowed, but usually not desirable.

3. A manager which decides gear shifting actions. At each decision time, the manager will act on a constrained set of gear shifting events to decide whether a gear shift should be commanded. The constrained set is determined by the following criteria: (1) Discrete constraints: This comes from the discrete event model, which eliminates several gear shifting actions such as 1-4, due to physical constraints. (2) Hybrid constraints: There are certain lower limits on engine speeds, called lugging limits, for gear shift decisions to avoid excessive harshness and vibration. These are hybrid limits since they depend on both analog output (vehicle speed) v and discrete state (gear) s .

3 Hybrid Control of Powertrain Automotive Systems

The current production strategy for powertrain control employs static mapping tables to schedule throttle and spark, and throttle-speed-gear operating points for gear shifting. For example, gear shifting from the second gear to the third gear will occur if the vehicle speed and throttle position move across a pre-calibrated line on the speed-throttle space.

The hybrid gear strategy developed in this paper employs feedback linearization and robust H^∞ design to derive an analog strategy for the throttle control, and a performance index to guide gear switching decisions. Details will be provided in the subsequent subsections. Generally speaking, this is the same idea as the performance guided robust hybrid control strategy introduced in [4]. Under some theoretical conditions, the generic strategy of [4] is shown to maintain robust stability and robust performance.

In applying this generic strategy to this case study, we have encountered some severe practical constraints which lead to several important modifications. Due to such modifications, the clean theoretical results of [4] do not hold rigorously in this case study. However, our simulation results still demonstrate similar stability, performance improvements, and robustness, to those claimed in [4].

3.1 Constraints and Strategy Modifications

Two-Time-Scale Strategy It is noted that the goal of analog control differs from that of discrete actions. To react promptly to the adverse effects of random disturbances on the system, the analog control must perform quickly. On the other hand, due to transient behavior of gear shifting and the desire to avoid gear hunting, discrete actions should have limited speed. As a result, we employ a two-time-scale method in this case study. In this method, one defines two time scales, T and T_d , where T is the sampling interval for analog control and T_d is

the discrete decision interval. T is much smaller than T_d . Usually, $T_d = k_d T$ for some integer, say, $k_d = 10$.

As a result, gear shifting decisions will only be made at kT_d for $k = 0, 1, 2, \dots$. In the time window $[kT_d, (k+1)T_d]$, an analog robust controller is employed to robustly stabilize the system, reduce the deviation of the vehicle speed from the driver's command speed, and achieve satisfactory fuel consumption. Analog control is then discretized with sampling interval T .

Real-Time Computation and Short Time Prediction Powertrain control strategies are to be implemented on-line. Due to the limited on-board computational resources, they must be computationally very efficient. The performance guided switching control of [4] computes first the expected worst-case future performance for each discrete state, and then selects the one offering the best future robust performance. While there are cases where such future robust performances can be explicitly computed [4] [8] [9], in automotive powertrain applications such computation is not only overly time consuming but impossible since the future commands from the driver are not available. A tradeoff, which is used in this study, is to replace the future performance index by its short horizon approximation. In this case, the performance measure is evaluated approximately only up to a short step into the future. For our simulation, the step is $2T_d$, where T_d is the interval for discrete decisions.

Actuator Saturation The throttle angle is saturated near 90 degrees. Also, the control authority of the throttle angle on the air charge depends on the manifold pressure. When the manifold pressure is close to atmospheric pressure 100 (KPa) that corresponds to the wide-open throttle operation, the throttle will lose control authority. In both cases, only the gear shifting remains a control variable. We have modified the hybrid control strategy such that it is reduced to discrete feedback only when either the throttle is WOT (wide open throttle) or the manifold pressure is close to the atmospheric pressure.

Unquantified Modeling Errors Although it is well understood that second-order models of engine systems are subject to significant modeling errors, due to data fitting errors, limited test data, engine-to-engine deviations and variations in operating conditions, the modeling errors cannot be well quantified. As a result, computation of the worst-case performance against modeling errors becomes difficult. On the other hand, a selection of the switching penalty matrix defines the level of robustness against modeling errors. In this case study, we select a switching penalty matrix to balance robustness, performance, and gear shifting frequency. Then simulation is performed to evaluate the design.

3.2 Analog Control Design

During the time interval between discrete decision instances, discrete states (gear) are unchanged and the analog control becomes the sole control authority. Apparently, in this case, the analog control must deliver the usual robust performance in the presence of modeling errors, time delays and disturbances.

The analog control employed here is a nonlinear robust H^∞ controller which is constructed according to the design method developed in [5]. The design method involves the following steps.

1. Feedback Linearization

Suppose the desired vehicle speed profile is given by $v_d(t)$. Note that for vehicles equipped with electronic throttles, the profile v_d is interpreted as the driver's pedal positions. In the case of the manual throttle operated by the driver, the profile $v_d(t)$ is the desired profile perceived by the driver.

The second-order powertrain nominal model is feedback linearizable and all the functions used in feedback linearization can be explicitly derived.

Define $z_1 = v - v_d$, $z_2 = f_v(p, v, i)$. Then, we have $\dot{z}_1 = z_2 - \dot{v}_d$ and

$$\begin{aligned}\dot{z}_2 &= \frac{\partial f_v}{\partial v} \dot{v} + \frac{\partial f_v}{\partial p} \dot{p} \\ &= \frac{\partial f_v}{\partial v} z_2 + \frac{\partial f_v}{\partial p} (f_p(p, v, i) + u) \\ &:= w.\end{aligned}$$

As a result, the nominal nonlinear system is transformed by a state mapping $(z_1, z_2) = T(p, v, i)$ and control mapping $u = \phi(p, v, i, w)$ to a linear system. It is noted, however, that modeling errors will be mapped into the new system as nonlinear uncertainties.

2. Design of the State Feedback

Now, a linear state feedback K can be constructed by employing the Riccati inequality approach developed in [5]. Numerically, this can be easily done by using Matlab functions. The system model in this case is a linear nominal part with nonlinear uncertainties. The modeling errors are given by $\|\Delta(x, u)\| \leq \varepsilon_1 \|x\| + \varepsilon_2 \|u\|$ where ε_1 and ε_2 are error bounds. The linearized nominal system is simply $\dot{z} = Az + B_1 w + B_2 d_w$ and $y = Cz$, where

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}; \quad B_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}; \quad B_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \quad C = [1 \ 0].$$

Note that the term d_w includes \dot{v}_d as well as other possible disturbances. The performance criterion is the H^∞ -type performance

$$\int_0^t (y^T W_y y + w^T W_w w) dt \leq \gamma^2 \int_0^t d_w^T W_d d_w dt, \quad \forall t.$$

It was shown in [5] that a sufficient condition for the performance criterion to be satisfied is that the Riccati Equation

$$A^T P + P A - P[B_1(W_w + \frac{\varepsilon_2}{\varepsilon_b})^{-1}B_1^T - \frac{1}{\gamma^2}B_2W_d^{-1}B_2 - (\varepsilon_1\varepsilon_a + \varepsilon_2\varepsilon_b)]P + [C^TW_yC + \frac{\varepsilon_1}{\varepsilon_a}I] = 0$$

has a positive definite solution P , where ε_a and ε_b are positive constants selected by the designer. Finally the state feedback is given by

$$K = -(W_w + \frac{\varepsilon_2}{\varepsilon_b})^{-1}B_1^T P.$$

3. Control Signal Construction

The overall analog control is constructed as follows: At time t , $v(t)$ and $p(t)$ are measured. Then $z_1(t) = v(t) - v_d(t)$ and $z_2(t) = f_v(p(t), v(t), i(t))$ are computed. After applying the linear state feedback $w(t) = K[z_1(t), z_2(t)]^T$, we obtain the original control from

$$u(t) = \frac{w(t) - \frac{\partial f_v}{\partial v} z_2}{\frac{\partial f_v}{\partial p}} - f_p(p(t), v(t), i(t)).$$

The design procedure has been applied to automotive engine control problems and demonstrates robustness against significant modeling errors, disturbances and engine delays [6].

3.3 Priority Functions and Switching Decisions

In this powertrain control problem, gear shifting is the discrete decision which must be made by a manager. Due to the requirements of on-line implementation, the optimization problem must be solved in a short period of time and solutions must be updated frequently. This forbids optimization over a large window of time interval. The decision-making process is based on a short-term performance index which contains three terms

$$J = W_a J_a + W_f J_f + W_s J_s$$

where J_a is a term on tracking performance, J_f is a measure of fuel consumption, and J_s reflects switching penalties. W_a, W_f, W_s are weightings to reflect tradeoffs among these competing performance objectives. Essentially, $J_a = \int_t^{t+2T_d} |v(\tau) - v_d(\tau)| d\tau$, $J_f = \int_t^{t+2T_d} \frac{air(\tau)}{14.6} d\tau$, and J_s is given by a 4×4 matrix whose (i, j) -th element is the switching penalty for gear shifting from position i to position j . This decision is made at kT_d , $k = 0, 1, 2, \dots$

4 Simulation Results

Substantial simulation has been performed to evaluate the hybrid control strategies, under various conditions. Selections of parameters for simulation are first provided.

Sampling interval for analog control $T = 0.05$ second. Discrete decision interval $T_d = 0.5$ second. Weighting values for gear shifting: $W_a = 400$, $W_f = 400$, $W_s = 5$. Gear shifting penalty is defined by a matrix SP whose (i, j) -th component is the switching penalty for shifting from gear i to gear j .

$$SP = \begin{bmatrix} 0 & 1 & 50 & 10000 \\ 10 & 0 & 5 & 100 \\ 100 & 10 & 0 & 10 \\ 10000 & 10 & 5 & 0 \end{bmatrix}$$

Note that the large value 10,000 entered for shifting from 1 to 4 or vice versa essentially excludes these shifts. If such shifts are physically forbidden, one can simply enter a much larger value to prevent such discrete decisions. Vehicle loads from road conditions are given by $P_L = a + bv^2$, where $a = 2.011 \times 10^{-3}$, and $b = 1.5 \times 10^{-6}$ for flat roads and $b = 2.65 \times 10^{-6}$ for hilly roads. External disturbances and modeling errors are added to the model in simulation.

1. Robustness and performance of switching control. The main purpose of this test is to evaluate the robustness, stability and performance of analog controllers for each fixed gear position, in the presence of modeling errors, time delays, disturbances. In Figures 5, 6, and 7, the robust analog controllers, together with the hybrid switching decision on gear selections, demonstrate robustness, stability and performance. For comparison purposes, we combined the same robust analog controllers with a typical production gear-shifting strategy. Similar robust performances are observed in Figure 4.

2. Different road and acceleration conditions.

Due to system nonlinearity, powertrain system performance varies significantly with vehicle loads and acceleration profiles. For instance, vehicle loads become much higher on a hilly road than on a flat road. In this study, we tested system performance under various road conditions including flat surfaces and hilly roads, as well as a combination of both. We also tested system performance under various acceleration commands, including fast and slow driving profiles. Figures 5, 6, and 7 show that the system performs very well under these various conditions using a robust controller with a hybrid gear strategy.

3. Comparison to typical production gear shifting strategies.

While it is quite obvious that hybrid control methodologies offer clearer understanding of design tradeoff and more systematic development tools, comparison is necessary to evaluate if the hybrid control actually offers any

advantages over the traditional production gear strategies in their performances. For this purpose, a typical and current production gear shifting strategy is employed, whose performance is depicted in Figure 4, to compare to the performance of the hybrid gear strategy in Figure 5. With a reasonable selection of weighting values, the hybrid strategy outperforms the production gear strategy in our simulation.

4. Manual versus electronic robust throttle controller.

To compare the electronic robust controller with a driver, we model a driver's manoeuvre of the foot-pedal in response to the vehicle speed by a PI controller where aggressive drivers might be modelled by using larger values K_p and K_I in the PI controller. For this simulation, $K_p = 2$ and $K_I = 0.05$. Performance of this PI controller is illustrated in Figures 8 and 9. While test results show that the robust controller (Figures 4 and 5) offers better overall performance than the PI controller in tracking the desired vehicle speed, it should be cautioned that the comparison is limited by our modeling of the driver's behavior which might be much more complicated than a PI controller.

5. Effects of weighting functions and switching penalty on system robustness and performance.

We used simulation to gain an understanding on roles played by the weighting matrices and potential guidelines in selecting such matrices. The following observations, which are to be expected, were made regarding the effects of the following parameters: As W_a increases, the tracking of speed is better, but fuel consumption becomes higher, as shown in Figures 5 and 10. On the other hand, as W_f increases, one observes an improvement on fuel consumption, at the expense of worse performance in speed tracking as shown in Figures 5 and 11. Finally, as W_s increases, infrequent gear shifting occurs and therefore causes higher fuel consumption as shown in Figure 12.

References

1. P.R. Crossley and J.A. Cook, IEE International Conference Control 91, Conference Publication 332, Vol. 2, pp. 921-925, Edinburgh, U.K, March 1991.
2. The Simulink Model, developed by Ken Butts, Ford Motor Company. Modified by Paul Barnard and Ted Liefeld, Mathworks, Inc.
3. L.Y. Wang, A. Beydoun, J. Cook, J. Sun and I. Kolmanovsky, Optimal hybrid control with automotive applications, in *Logic-Based Switching Control*, A.S. Morse (ED.), Springer, 1996.
4. L.Y. Wang, P. Khargonekar and A. Beydoun, Structures and control of hybrid systems, *Hybrid Systems V*, Notre Dame, Indiana, September 1997.
5. L.Y. Wang and W. Zhan, Robust disturbance attenuation with stability for linear systems with norm-bounded nonlinear uncertainties, *IEEE Trans. Automat. Control*, Vol. AC-41, 1996, pp. 886-888.

6. W.J. Zhang, Robust stabiliztion and disturbance attenuation of nonlinear systems with applications to engine systems, Ph.D. dissertation, Wayne State University, 1997.
7. A.S. Morse (Ed.), Control Using Logic-Based Switching, Springer Verlag, LNCIS 222, 1996.
8. G.L. Grossman, A. Nerode, A.P. Ravn and H. Rischel (Eds.), Hybrid Systems, Springer Verlag, LNCS 736, 1993.
9. P. Antsaklis, W. Kohn, A. Nerode and S. Sastry (Eds.), Hybrid Systems II, Springer Verlag, LNCS 999, 1995.

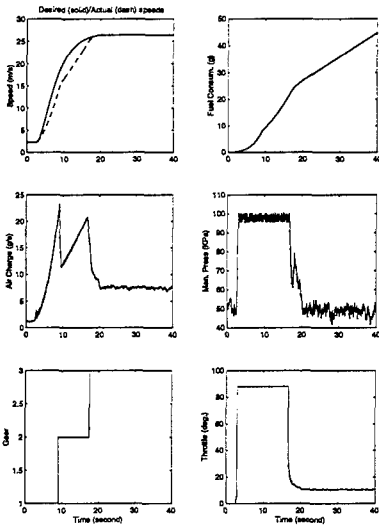


Fig. 4. Powertrain systems controlled by robust analog controller and production gear strategy under flat road condition.

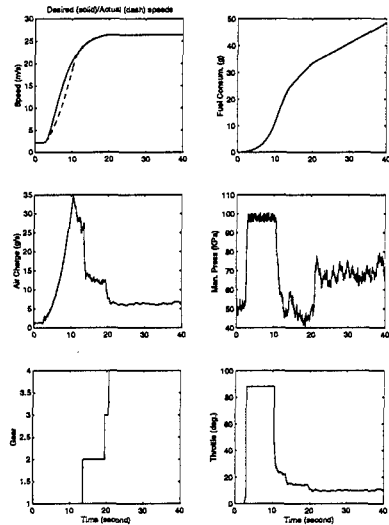


Fig. 5. Powertrain systems controlled by robust analog controller and hybrid gear strategy under flat road condition.

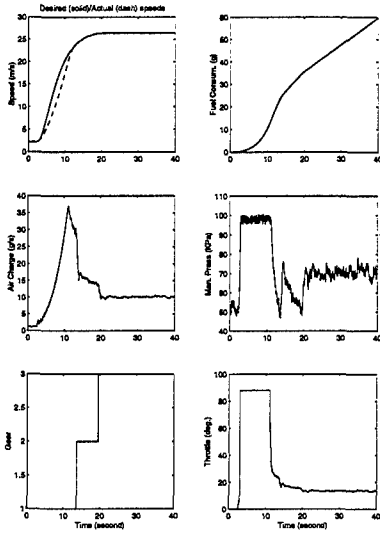


Fig. 6. Powertrain systems controlled by robust analog controller and hybrid gear strategy under hilly road condition.

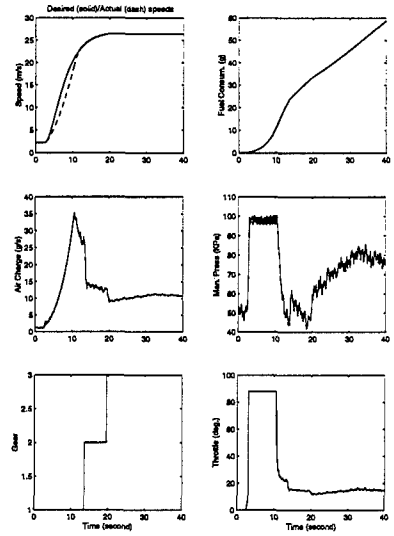


Fig. 7. Powertrain systems controlled by robust analog controller and hybrid gear strategy under combined road conditions.

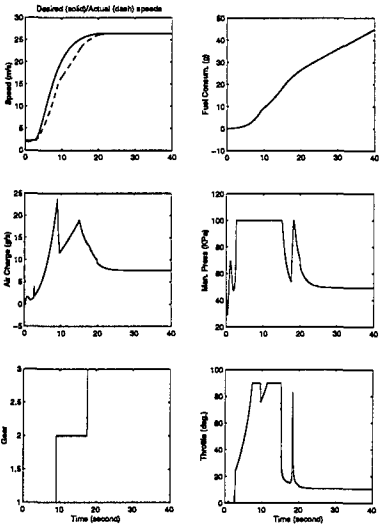


Fig. 8. Powertrain systems with a manual throttle and production gear strategy under flat road condition.

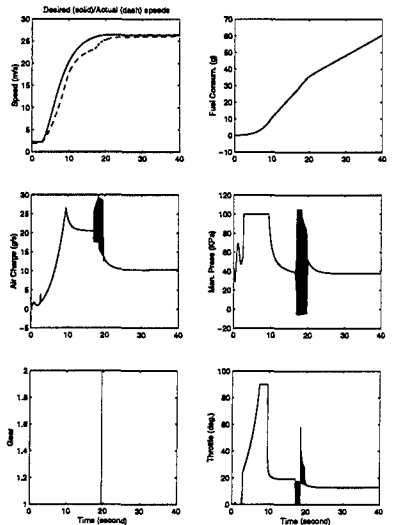


Fig. 9. Powertrain systems with a manual throttle and hybrid gear strategy under flat road condition.

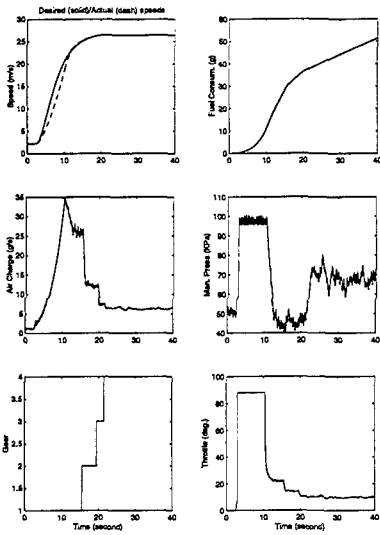


Fig.10. Powertrain systems controlled by robust analog controller and hybrid gear strategy under flat road condition. The weighting W_a is increased to 700.

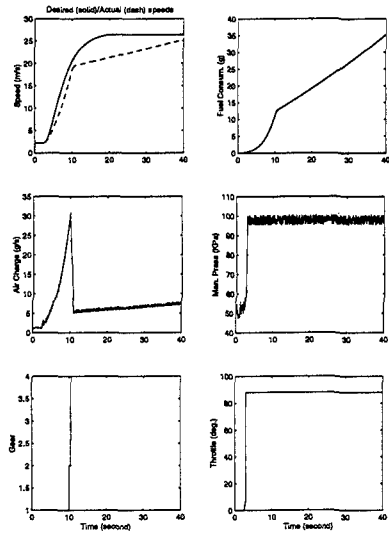


Fig.11. Powertrain systems controlled by robust analog controller and hybrid gear strategy under flat road condition. The weighting W_f is increased to 800.

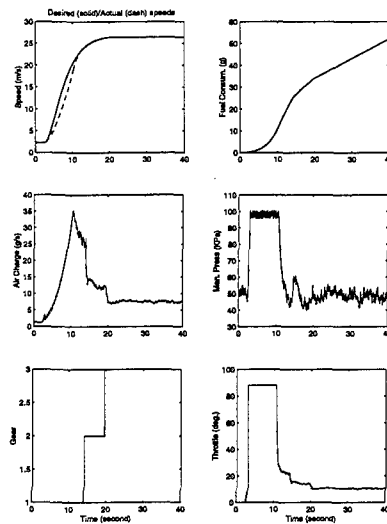


Fig.12. Powertrain systems controlled by robust analog controller and hybrid gear strategy under flat road condition. The weighting W_s is increased to 25.

On the Composition of Hybrid Systems

Sébastien Bornot and Joseph Sifakis
Sebastien.Bornot@imag.fr Joseph.Sifakis@imag.fr

VERIMAG, 2 rue Vignate, 38610 Gières, France

1 Introduction

Concurrent systems can be usually specified as systems of communicating processes obtained by composing sequential processes by means of binary parallel composition operators. The latter express process interaction in terms of action composition. Their semantics is usually defined by two types of rules.

- Synchronization rules that specify how an action of the product process is defined as the result of the (simultaneous) occurrence of two actions in two component processes.
- Interleaving rules, that specify how an action of a component process is an action of the product process. These rules allow some component processes to be idle while the others progress.

Combining synchronization and interleaving rules is essential for the specification of systems as process coordination requires both synchronization and waiting. However, their adequate combination must satisfy two conflicting requirements :

Deadlock-freedom : Deadlocks may appear in the product process as a result of enforcing synchronization, for instance, when two processes are at states from which only non matching synchronization actions can be performed. Such deadlocks can be avoided by using “escape” transitions generated by application of interleaving rules. However, the presence of both synchronization and interleaving actions may imply non maximal progress.

Maximal progress : When synchronization of two actions is possible, interleaving rules, used precisely to avoid deadlocks, may be applicable. Maximal progress means that synchronization is preferred to interleaving when both are possible. This is sometimes achieved by using restriction or hiding operators that prune out interleaving actions.

The above problems are amplified for timed or hybrid systems where time progress is synchronous and waiting times are bounded. This can be easily observed when hybrid specifications are obtained by adding timing constraints to untimed communicating systems specifications, as it has been pointed out in [SY96].

In [SY96,BS97b] it is claimed that specifying time progress conditions independently from discrete transitions may be source of inconsistencies in specifications. We propose a model where time progress constraints are associated with

actions and thus time progress is directly related with the ability of a system to perform actions. This model satisfies the property of *time reactivity* in the sense that if no action is enabled at a state, time can progress.

Following the process algebra approach, we consider discrete (untimed) systems represented as terms generated from a set of abstract actions by using operators such as prefixing, non deterministic choice and parallel composition. We extend the semantics of these operators to hybrid actions.

For a given abstract action a , a *hybrid action* extension of a , is defined as a triple (g_a, d_a, f_a) where g_a and d_a are unary predicates and f_a is a total function on a continuous set of states. The predicate g_a is a *guard* characterizing the states from which a is enabled while d_a is a *deadline* satisfied by all the enabling states at which the action a becomes urgent (time progress is stopped). The function f_a represents the effect of the action when it is executed.

As usually, for a given n -ary operator op , the hybrid actions of the term $op(t_1, \dots, t_n)$ are obtained by composing the hybrid actions of the arguments t_i . We show that the semantics of operators on abstract actions can be extended to hybrid actions in different manners. The extensions have the same semantics for discrete transitions but may differ in urgency (ability to perform actions within a given delay).

We assume that parallel composition of two discrete systems can be expressed as the non-deterministic choice of terms starting with interleaving or synchronization actions (by means of some expansion theorem [BK85]). The expansion theorem is extended to hybrid actions in the following manner :

- To guarantee maximal progress, non-deterministic choice is replaced by priority choice that gives higher priority to synchronization actions over interleaving actions.
- Synchronization operators between abstract actions are extended to hybrid actions. The guard and the deadline resulting from the synchronization of two hybrid actions depend on the guards and deadlines of the synchronizing hybrid actions. We show that for hybrid actions different synchronization operations of practical interest can be defined by taking as synchronization guards and deadlines modal formulas. In particular, we identify three important synchronization modes : AND-synchronization where the guards of the synchronization action is the conjunction of the guards of the contributing actions. MAX-synchronization used to model synchronization with waiting and for which the synchronization action occurs as soon as all of the contributing actions have been completed. MIN-synchronization where the synchronization action occurs as soon as one of the contributing actions is completed.

The paper is organized as follows. In section 2, we define hybrid extensions of discrete systems as a labeling homomorphism that extends prefixing and choice operators. Section 3 presents a framework for parallel composition of hybrid systems as an extension of parallel composition of untimed systems. For the

three basic synchronization modes parallel composition rules are proposed that guarantee both local deadlock-freedom and maximal progress. We conclude by indicating possible application directions.

2 Hybrid extensions of discrete systems

We consider a simple (discrete) algebra of terms S_A with prefixing and non-deterministic choice. We show that a hybrid extension of S_A can be defined as a labeling of the underlying transition system associating with a state s , an evolution function \triangleright_s and with any action a a hybrid action $h(a)$.

2.1 Discrete systems

Consider the language of terms S_A defined by the grammar

$$s ::= Nil \mid a.s \mid s + s$$

where Nil is a constant and $a \in A$, a set of atomic actions.

With a term of S_A we associate transition relations subsets of $S_A \times A \times S_A$ defined by

$$\begin{aligned} a.s &\xrightarrow{a} s \\ s_1 &\xrightarrow{a} s_1' \text{ implies } s_1 + s_2 \xrightarrow{a} s_1' \text{ and } s_2 + s_1 \xrightarrow{a} s_1' \end{aligned}$$

We consider that $+$ is an associative commutative operator with Nil as zero element. Any term s is congruent (strongly bisimilar) to a term of the form :

$$s = \sum_{i \in I} a_i.s_i \quad (\text{taken to be } Nil \text{ if } I = \emptyset)$$

2.2 Hybrid extension of S_A

A hybrid extension of S_A is defined as a pair (V, h) where

- V is a continuous state space isomorphic to \mathbf{R}^n for some $n > 0$
- h is a labeling of S_A such that :
 - $h(s) = (s, \triangleright_s)$, where $\triangleright_s : V \times \mathbf{R}_+ \rightarrow V$ is an *evolution function*. We write $v \triangleright_s t$ for $\triangleright_s(v, t)$. We require that \triangleright_s is *additive*, i.e., $\forall v \in V \forall t_1, t_2 \in \mathbf{R}_+. v \triangleright_s (t_1 + t_2) = (v \triangleright_s t_1) \triangleright_s t_2$.
 - $h(a) = (a, g, d, f)$ where g and d are two unary predicates on V and $f : V \rightarrow V$. We suppose that $d \Rightarrow g$. We call g, d, f the *guard*, the *deadline* and the *jump* respectively of the *hybrid action* $h(a)$ associated with a .

The hybrid extension of the term $s = \sum_i a_i.s_i$ is represented by the term $h(s) = \sum_i h(a_i).h(s_i)$.

We define hereafter the semantics of $h(s)$ in two steps. First, we associate transition relations with hybrid actions $h(a_i)$ on the continuous state space V . Then, we define the transition relation of the hybrid extension.

Definition 1. Let $b = (a, g, d, f)$ be a hybrid action associated with a in some transition $s \xrightarrow{a} s'$ of S_A . We define transition relations \xrightarrow{t} for $t \in \mathbf{R}_+$ and \xrightarrow{a} for $a \in A$ subsets of $V \times V$:

- $b : v \xrightarrow{t} v \triangleright_s t$ if $\forall t' < t. \neg d(v \triangleright_s t')$
- $b : v \xrightarrow{a} f(v)$ if $g(v)$

The two relations describe the behavior of b from a continuous state v . $b : v \xrightarrow{t} v \triangleright_s t$ means that the execution of b can be delayed for t time units and $b : v \xrightarrow{a} f(v)$ represents the effect of a jump.

Definition 2.

The semantics of $h(s) = \sum_i b_i \cdot h(s_i)$ where $b_i = (a_i, g_i, d_i, f_i)$ and $h(s) = (s, \triangleright_s)$ is defined as a family of labeled transitions, subsets of $(S_A \times V) \times (A \cup \mathbf{R}_+) \times (S_A \times V)$ by the rules

- If $b_i : v \xrightarrow{a_i} v_i$ then $(s, v) \xrightarrow{a_i} (s_i, v_i)$
- If $\forall i \in I. b_i : v \xrightarrow{t_i} v \triangleright_{s_i} t_i$ then $(s, v) \xrightarrow{t} (s, v \triangleright_s t)$.

Remark 3.

Notice that the projection of the transition relations on discrete state components agrees with the transition relations of the associated discrete system. This justifies the use of the term “extension”.

Time can advance in $h(s)$ for $s = \sum_i a_i \cdot s_i$ only if all the hybrid actions $h(a_i)$ agree to let time advance. This rule determines a time progress condition associated with s similar to the “invariants” in [ACH⁺95] and “time progress conditions” in [KMP96]. Associating time progress with actions is an important feature of the presented model as it will be shown throughout the paper. For a given hybrid action, its guard characterizes the states from which the action is possible while its deadline characterizes the subset of the states where the action is enforced by stopping time progress.

The condition $d \Rightarrow g$ guarantees that if no action is enabled from a state then time can progress. In fact, time progress can stop only at states where a guard is enabled. Using terminology from synchronous language [JM94] we call this property *time reactivity*.

The relative position of d with respect to the corresponding g determines the *urgency* of an action. For a given g , the corresponding d may take two extreme values: $d = g$ which means that the action is *eager* and $d = \text{false}$ which means that the action is *lazy*. A particularly interesting case is the one of *delayable* action where d is the falling edge of g (cannot be disabled without enforcing its execution) (figure 1).

2.3 Choice operators

Let $B = \{b_i\}_{i \in I}$ be a set of actions $b_i = (a_i, g_i, d_i, f_i)$ labeling transitions issued from a term with evolution function \triangleright . We use the modal operators $\Diamond_{\leq k} p$

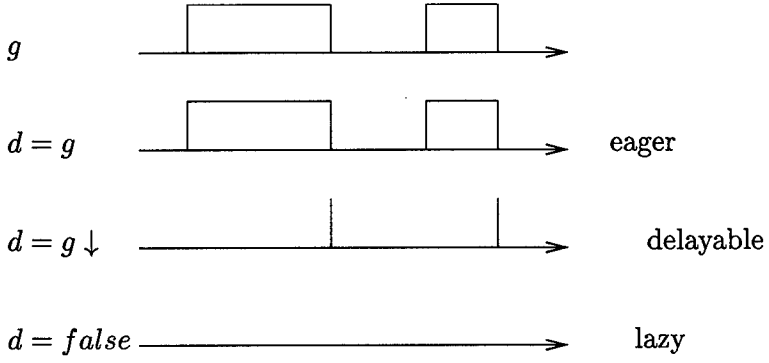


Fig. 1. using deadlines to specify urgency

(eventually p within k) and $\Diamond_{\leq k} p$ (once p since k) where p is a unary predicate on V , and $k \in \mathbf{R}_+ \cup \{\infty\}$.

$$\begin{aligned} \Diamond_{\leq k} p(v) &\text{ if } \exists t \in \mathbf{R}_+ \ 0 \leq t \leq k. p(v \triangleright t) \\ \Diamond_{\leq k} p(v) &\text{ if } \exists t \in \mathbf{R}_+ \ 0 \leq t \leq k. \exists v' \in V. v = v' \triangleright t \wedge p(v') \end{aligned}$$

As usual, we write $\Diamond p$ and $\Diamond p$ for $\Diamond_{\leq \infty} p$ and $\Diamond_{\leq \infty} p$ respectively, and $\Box p$ and $\Box p$ for $\neg \Diamond \neg p$ and $\neg \Diamond \neg p$ respectively.

We have already defined a *non-deterministic* choice operator $\sum_i b_i.s_i$ which combines the semantics of hybrid actions in a very simple manner. The discrete transition relation is the union of the discrete transition relations of the hybrid actions b_i and the timed transition relation is the intersection of the timed transition relations of the b_i 's. This semantics corresponds to a maximally urgent behavior in the sense that an action may occur when $\bigvee_i g_i$ holds and time progress stops as soon as $\bigvee_i d_i$ holds. In practice, it is often useful to define other choice operators with less prompt semantics ([BS97a]). We define a choice operator taking into account priorities between actions. Instead of considering non-deterministic choice between actions $b_i = (a_i, g_i, d_i, f_i)$, for $i = 1, 2$, one can consider that, for instance, b_2 has higher priority than b_1 which leads to restricting the guard and the deadline of b_1 to g_1' and d_1' respectively. One may take $g_1' = g_1 \wedge \neg g_2$ and $d_1' = d_1 \wedge g_1'$ to resolve conflicts between b_1 and b_2 in favor of b_2 . This is a well-known manner to give priority to actions in untimed systems. However, for timed systems priority can concern not only instantaneous conflict resolution but also take into account possibility of waiting. For instance, if we take $g_1' = g_1 \wedge \Box \neg g_2$ and $d_1' = d_1 \wedge g_1'$, we restrict the enabling states of b_1 to only those states from which b_2 will never be enabled.

Definition 4. *priority order*

Consider the relation $< \subseteq A \times (\mathbf{N} \cup \{\infty\}) \times A$. We write $a_1 <_k a_2$ for $(a_1, k, a_2) \in <$ and suppose that

$<_k$ is a partial order relation for all $k \in \mathbb{N} \cup \{\infty\}$

$a_1 <_k a_2 \Rightarrow \forall k' < k. a_1 <_{k'} a_2$

$a_1 <_k a_2 \wedge a_2 <_l a_3 \Rightarrow a_1 <_{k+l} a_3$

Property : The relation $a_1 \ll a_2 = \exists k a_1 <_k a_2$ is an order relation.

Definition 5. priority choice operator

Given $<$, a priority order and $\{b_i.s_i\}_{i \in I}$, a set of term, we define the priority choice operator $\sum_{<}$ such that :

$$\sum_{<} \{b_i.s_i\}_{i \in I} = \sum_{i \in I} b'_i.s_i$$

where if $b_i = (a_i, g_i, d_i, f_i)$ then $b'_i = (a_i, g'_i, d'_i, f_i)$ with $g'_i = g_i \wedge \bigwedge_{a_i <_k a_j} \neg \Diamond_{\leq k} g_j$ and $d'_i = d_i \wedge g'_i$.

Notice that if $a_i <_k a_j$ then in $\sum b'_i.s_i$ " a_j has higher priority than a_i in the interval $[0, k]$ " that is, a_i is disabled if a_j will be enabled within k time units.

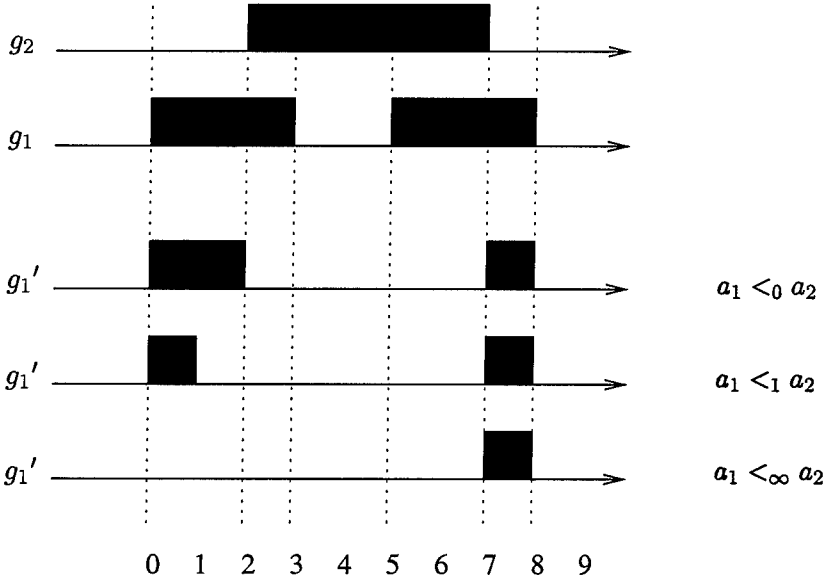


Fig. 2. Different priorities for a_2 over a_1

Consider the guards g_1, g_2 of the actions a_1, a_2 . Figure 2 gives the guards g'_1 obtained when g_1 is restricted by considering the priority orders $a_1 <_0 a_2$, $a_1 <_1 a_2$, $a_1 <_\infty a_2$.

Proposition 6. *The priority choice operators defined above satisfy the following properties.*

1. $\Diamond g_i \Rightarrow \Diamond(g'_i \vee \bigvee_{a_i \ll a_j} g_j)$
2. $\Diamond \bigvee_{i \in I} g_i = \Diamond \bigvee_{i \in I} g'_i$

The first property means that if action a_i can occur in the non-prioritized choice then either a_i can occur in the prioritized choice or some action of higher priority.

The second property is a consequence of the first and simply says that $\sum_{<}$ preserves (local) deadlock-freedom : if some action can be executed in the non-prioritized choice then some action can be executed in the prioritized choice and vice versa.

3 Parallel composition

In this section we define parallel composition operators by following the same approach as in the previous section. First, we show how parallel composition on hybrid systems can be defined as an extension of parallel composition on untimed systems. We thus obtain general composition rules for which some practically interesting cases are discussed later.

3.1 Extending parallel composition from untimed to hybrid systems

Untimed systems We consider a general framework for the composition of untimed terms. For this, we suppose that the vocabulary of actions A contains a distinguished element \perp and consider the set $A^!$ of the words generated from A with a commutative operator $!$ such that for all a , $a! = \perp$. The operator $!$ is usually called communication function [BK85]. The words are used to represent synchronization actions that is, actions that result from the synchronous occurrence of atomic actions. $a_1!a_2 = \perp$ means impossibility of synchronization.

In the sequel, we suppose that there are no other simplification rules for $!$ but the rule for \perp and that a word $a_i!a_j$ is given in reduced form.

Consider the language of terms $S_{A^!}$ defined by the grammar

$$s ::= s \in S_A \mid s \parallel s$$

The semantics of the parallel composition operator is defined by the rules

$$\left\{ \begin{array}{l} s_1 \xrightarrow{a_1} s_1' \\ s_2 \xrightarrow{a_2} s_2' \end{array} \right\}, a_1!a_2 \neq \perp \text{ implies } \left\{ \begin{array}{l} s_1 \parallel s_2 \xrightarrow{a_1!a_2} s_1' \parallel s_2' \\ s_2 \parallel s_1 \xrightarrow{a_2!a_1} s_2' \parallel s_1' \end{array} \right.$$

$$s_1 \xrightarrow{a_1} s_1' \text{ implies } \left\{ \begin{array}{l} s_1 \parallel s_2 \xrightarrow{a_1} s_1' \parallel s_2 \\ s_2 \parallel s_1 \xrightarrow{a_1} s_2 \parallel s_1' \end{array} \right.$$

\parallel is a commutative operator that can be expressed in terms of non-deterministic choice. It is well-known that for $q_1 = \sum_i a_i.s_i$ and $q_2 = \sum_j a_j.s_j$,

$$q_1 \parallel q_2 = \sum_i a_i.(s_i \parallel q_2) + \sum_j a_j.(s_j \parallel q_1) + \sum_{i,j} a_i a_j.(s_i \parallel s_j)$$

The first two summands start with interleaving actions while the last one starts with synchronization transitions (only terms such that $a_i a_j \neq \perp$ appear).

Hybrid extension of S_{A_i} For given (V_i, h_i) hybrid extensions of q_i for $i = 1, 2$, a hybrid extension (V, h) for $q_1 \parallel q_2$ is defined by :

- $V = V_1 \times V_2$
- If $\tau_i = s_i \xrightarrow{a_i} s_i'$ is a transition of q_i then $q_1 \parallel q_2$ has transitions of the form $\tau = s_1 \parallel s_2 \xrightarrow{\lambda} s_1' \parallel s_2'$ where $\lambda = a_i$ or $\lambda = a_1 a_2$. We take $h(\tau) = (s_1 \parallel s_2, \triangleright_{s_1} \times \triangleright_{s_2}) \xrightarrow{h(\lambda)} (s_1' \parallel s_2', \triangleright_{s_1'} \times \triangleright_{s_2'})$ where
 - $h(\lambda) = h_i(a_i)$ if $\lambda = a_i$ and $h(\lambda) = h_1(a_1)h_2(a_2)$ if $\lambda = a_1 a_2$ (we extend the communication function in an appropriate manner to hybrid actions, see below).
 - $\triangleright_{s_1} \times \triangleright_{s_2} : (V_1 \times V_2) \times \mathbf{R}_+ \rightarrow V_1 \times V_2$ is such that $(v_1, v_2)(\triangleright_{s_1} \times \triangleright_{s_2})t = (v_1 \triangleright_{s_1} t, v_2 \triangleright_{s_2} t)$.

This definition leads, by taking $b_i = h_1(a_i)$ and $b_j = h_2(a_j)$, to a scheme of expansion theorem for parallel composition where \oplus and \bigoplus are arbitrary choice operators (as defined in the previous section and in [BS97a]) :

$$\begin{aligned} h(q_1 \parallel q_2) &= h_1(q_1) \parallel h_2(q_2) = \sum_i b_i.h_1(s_i) \parallel \sum_j b_j.h_2(s_j) \\ &= \bigoplus_i b_i.(h_1(s_i) \parallel \sum_j b_j.h_2(s_j)) \oplus \bigoplus_j b_j.(h_2(s_j) \parallel \sum_i b_i.h_1(s_i)) \\ &\quad \oplus \bigoplus_{i,j} (b_i b_j).(h_1(s_i) \parallel h_2(s_j)) \end{aligned}$$

If \oplus and \bigoplus are non-deterministic choice operators then maximal progress is not guaranteed as an interleaving action may be executed when synchronization is possible. For this reason, we define parallel composition as the priority choice of the expanded terms with infinite priority to synchronization actions $b_i b_j$ over the interleaving actions b_i and b_j . This corresponds to priority choice for the minimal order $<$ such that $a_i <_{\infty} a_{i,j}$ and $a_j <_{\infty} a_{i,j}$ for any i, j . By using the notations

$$\begin{aligned} B &= \{b_i.(h_1(s_i) \parallel \sum_j b_j.h_2(s_j))\}_i \cup \{b_j.(h_2(s_j) \parallel \sum_i b_i.h_1(s_i))\}_j \\ &\quad \cup \{(b_i b_j).(h_1(s_i) \parallel h_2(s_j))\}_{i,j} \end{aligned}$$

and $h_1(q_1) = \sum_i b_i.h_1(s_i)$ and $h_2(q_2) = \sum_j b_j.h_2(s_j)$, we have $h_1(s_1) \parallel h_2(s_2) = \sum_{<} B$ which is equivalent to

$$\sum_i b_i'.(h_1(s_i) \parallel h_2(s_2)) + \sum_j b_j'.(h_2(s_j) \parallel h_1(s_1)) + \sum_{i,j} b_i b_j.(h_1(s_i) \parallel h_2(s_j))$$

(figure 3)

In the above term, b_i', b_j' are the actions obtained by restricting b_i and b_j due to priority. We now define $b_i b_j$.

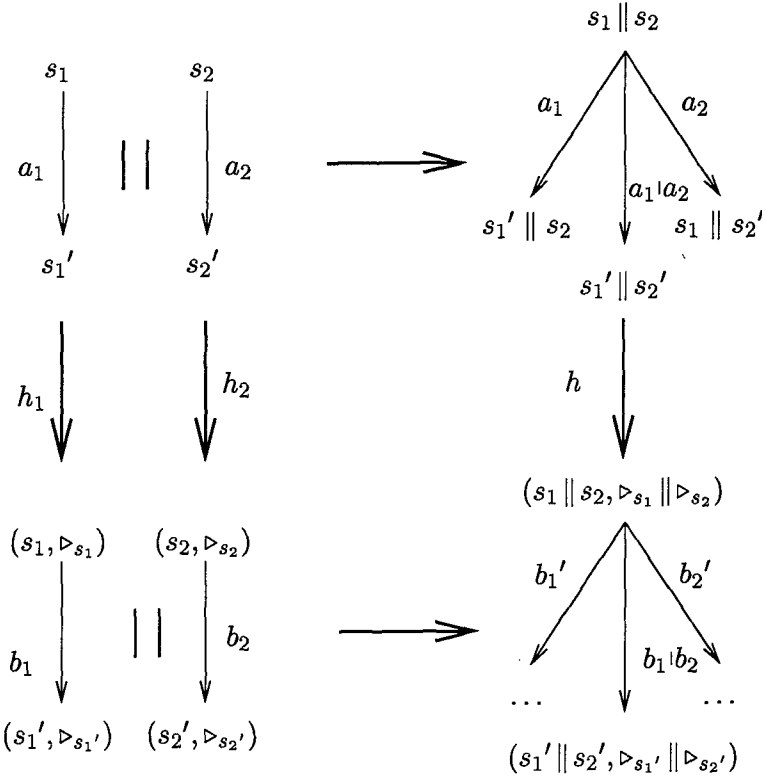


Fig. 3. Hybrid extension for parallel composition

Suppose that $h(a_i) = b_i = (a_i, g_i, d_i, f_i)$ for $i \in I$. If $a_i|a_j = \perp$ then we take $b_i|b_j = \perp$. Otherwise, we write $b_{i,j} = b_i|b_j = h(a_i|a_j) = h_1(a_i)|h_2(a_j) = (a_i|a_j, g_{i,j}, d_{i,j}, f_i \times f_j)$ where

$f_i \times f_j : V_1 \times V_2 \rightarrow V_1 \times V_2$ such that $(f_i \times f_j)(v_1, v_2) = (f_i(v_1), f_j(v_2))$.

We propose in the next subsection a method for defining $g_{i,j}$ and $d_{i,j}$ by respecting the requirements $g_{i,j} \Rightarrow g_i \vee g_j$ and $d_{i,j} \Rightarrow d_i \vee d_j$ which mean that $b_{i,j}$ may be caused only by b_i or b_j .

Proposition 7. *If $g_{i,j} \Rightarrow g_i \vee g_j$, the above definition guarantees the following properties*

1. local deadlock-freedom preservation that is,

$$\Diamond(\bigvee_{i \in I} g_i \vee \bigvee_{j \in J} g_j) = \Diamond(\bigvee_{i \in I} g'_i \vee \bigvee_{j \in J} g'_j \vee \bigvee_{i \in I, j \in J} g_{i,j})$$

2. maximal progress that is, interleaving actions are executed only if synchronizations $b_{i,j}$ are disabled forever.

It is important to notice that these properties hold independently of the way the guards and deadlines of the synchronization actions are defined.

3.2 Synchronization modes of hybrid actions

Given two hybrid actions b_1, b_2 we define the guard $g_{1,2}$ and the deadline $d_{1,2}$ of the hybrid action $b_1 \parallel b_2 = (a_1 \parallel a_2, g_{1,2}, d_{1,2}, f_{1,2})$ resulting from their appropriate synchronization.

Composition of guards : synchronization modes As already discussed in [SY96,BS97b], for timed and hybrid systems the guard $g_{1,2}$ can be in general a modal formula in terms of the guards g_1 and g_2 . We consider in particular three important synchronization modes :

AND-synchronization requires that synchronization takes place only when both synchronized transitions can be executed. This means $g_{1,2} = g_1 \wedge g_2$. Consider the example of two synchronizing actions with guards g_1 and g_2 . Then, in general interleaving actions are needed to avoid deadlock. Their guards in this case will be $g_1' = g_1 \wedge \Box \neg (g_1 \wedge g_2)$ and $g_2' = g_2 \wedge \Box \neg (g_1 \wedge g_2)$.

MAX-synchronization requires that the first of the two synchronized actions that becomes enabled awaits for the other to become enabled. The enabling of the latest action triggers synchronization. A consequence of this assumption is that waiting may be unbounded. For a given execution trace, the time interval in which the synchronized action is enabled has as lower bound the **maximum** of the times they become enabled and as upper bound the **maximum** of the times they become disabled. The corresponding guard $g_{1,2}$ is defined by

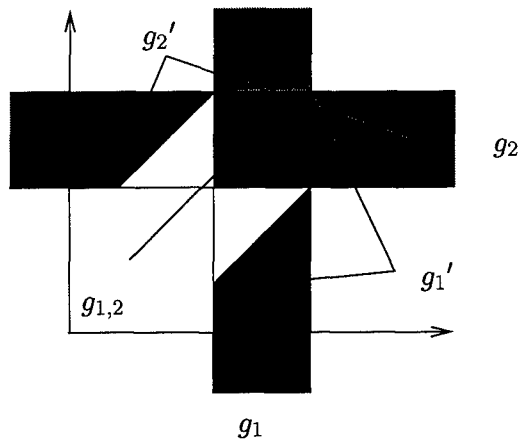


Fig. 4. AND-synchronization

$g_{1,2} = (\Diamond g_1 \wedge g_2) \vee (g_1 \wedge \Diamond g_2)$. For this condition to express synchronization with waiting, it is necessary that if s_1 and s_2 are the source states of the transitions labeled by b_1 and b_2 , these states should always be reached with values v_1 and v_2 such that $v_i \models_{s_i} \Diamond g_i$ (remember that the meaning of \Diamond depends of the evolution function \triangleright_{s_i}). In the case where there are only two synchronizing actions whose

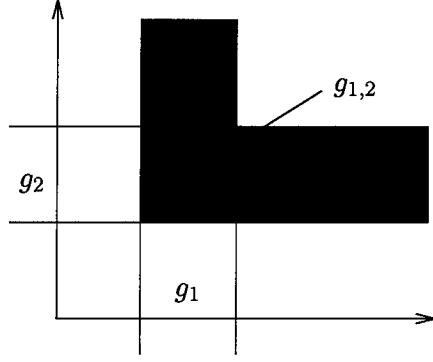


Fig. 5. MAX-synchronization

guards are g_1 and g_2 , the interleaving actions will have guards $g_1' = g_1 \wedge \Box \neg g_{1,2}$ and $g_2' = g_2 \wedge \Box \neg g_{1,2}$, which can be simplified into $g_1' = g_1 \wedge \Box \Box \neg g_2$ and $g_2' = g_2 \wedge \Box \Box \neg g_1$.

MIN-synchronization is the dual of the previous synchronization mode, and it implies that the synchronization action $a_1 \mid a_2$ can occur when one of the two synchronizing actions is enabled and the other will be eventually enabled. That is, synchronization may occur in a time interval whose lower bound is the **minimum** of the times they become enabled and the upper bound is the **minimum** of the times they become disabled. The corresponding guard $g_{1,2}$ is described by the formula $g_{1,2} = (\Diamond g_1 \wedge g_2) \vee (g_1 \wedge \Diamond g_2)$. In the case where

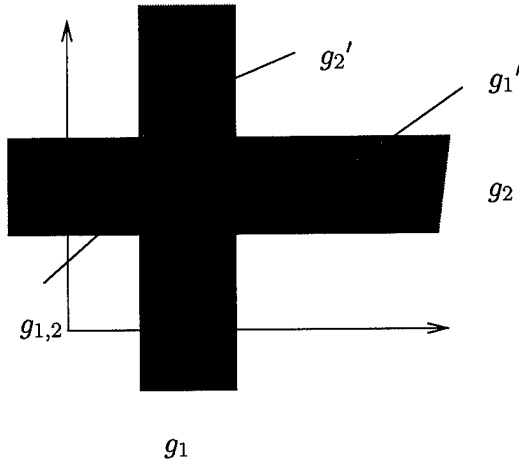


Fig. 6. MIN-synchronization

there are only two synchronizing actions with guards g_1 and g_2 , the interleaving actions will have guards $g_1' = g_1 \wedge \Box \neg g_{1,2} = g_1 \wedge \Box \neg g_2$ and $g_2' = g_2 \wedge \Box \neg g_1$.

Composition of deadlines : typed transitions For two given hybrid actions $b_1 = (a_i, g_i, d_i, f_i)$, $i = 1, 2$ the deadline $d_{1,2}$ corresponding to $b_1 b_2$ must satisfy the following condition

$$d_{1,2} \Rightarrow g_{1,2} \wedge (d_1 \vee d_2)$$

Of course, the most urgent solution is to take $d_{1,2} = g_{1,2} \wedge (d_1 \vee d_2)$ but this often leads to situations where the computed deadline $d_{1,2}$ does not correspond to the intuition [BS97a]. For this reason but also to introduce a simple model where deadlines are defined from guards by means of simple assumptions about urgency of the actions, we slightly modify our model.

We suppose that the deadline d_i of a hybrid action $b_i = (a_i, g_i, d_i, f_i)$ is defined by a function $\delta_i : 2^V \rightarrow 2^V$ such that $\delta_i(g_i) = d_i$.

An example of such a function is \downarrow (*falling edge*). When $d_i = g_i \downarrow$ we have a delayable action according to our terminology. Another example is the identity function $\mathbf{1} = \lambda g. g$ which can be used to define eager actions. Finally, a trivial case is the function $\mathbf{0} = \lambda g. \text{false}$ that allows to define lazy actions.

We call the function $\delta_i \in \{ \mathbf{0}, \downarrow, \mathbf{1} \}$ *types* of the action. Types characterize the urgency of an action which is minimal for $\mathbf{0}$ and maximal for $\mathbf{1}$. Clearly, for synchronization between b_1 and b_2 it is necessary to define $\delta_{1,2}$ such that

$$\boxed{\delta_{1,2}(g_{1,2}) \Rightarrow g_{1,2} \wedge (\delta_1(g_1) \vee \delta_2(g_2))} \quad (\alpha)$$

Proposition 8. *The following table gives the most urgent type $\delta_{1,2}$ satisfying (α) for any mode (AND, MAX, MIN) in terms of δ_1, δ_2 .*

δ_2	δ_1	$\mathbf{0}$	\downarrow	$\mathbf{1}$
$\mathbf{0}$		$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}$
\downarrow		$\mathbf{0}$	\downarrow	\downarrow
$\mathbf{1}$		$\mathbf{0}$	\downarrow	$\mathbf{1}$

This result allows to reason only in terms of types of actions and drastically simplifies the general framework.

To complete the results we show that the type of a transition is preserved by priorities and thus the type of interleaving actions is the same as the type of the corresponding synchronizing transitions.

Proposition 9. *If $d_i = g_i$ or $d_i = g_i \downarrow$ and $g_i' = g_i \wedge \Box \neg g$ for some g , then $d_i' = d_i \wedge g_i'$ is such that $d_i' = g_i'$ or $d_i' = g_i' \downarrow$ respectively.*

4 Applications

As an application of the above results, we define a parallel composition operator for typed hybrid actions that is, actions $b_i = (a_i, g_i, \delta_i, f_i)$ such that $\delta_i \in \{0, \downarrow, 1\}$.

We suppose that for each pair of actions (a_1, a_2) the synchronization mode is given. The resulting interleaving and synchronization actions depend on the synchronization mode. The synchronization action $b_{1,2}$ is $b_{1,2} = (a_1 a_2, g_{1,2}, \delta_{1,2}, f_{1,2})$ where $g_{1,2}$ is defined in 3.2 according to the synchronization mode and $\delta_{1,2}$ is as specified in the table given in 3.2. The interleaving actions b'_i are of the form $b'_i = (a_i, g'_i, \delta'_i, f_i)$ where $g'_i = g_i \wedge \Box \neg g_{1,2}$ and $\delta'_i = \delta_i$ (by proposition 9) for $i = 1, 2$.

Some applications of this general framework can be found in [SY96] where it is shown that for timed Petri nets the underlying synchronization mode is MAX-synchronization. This allows to represent state machine decomposable timed Petri nets as the MAX-parallel composition of timed automata with delayable actions and makes possible the application of efficient timing analysis techniques to timed Petri nets.

An application domain for our results is modeling of multimedia systems where combinations of the different synchronization modes are necessary for a natural description of timing constraints. Several formalisms used in this area offer such possibilities. One of the most general seems to be the model of Time Stream Petri Nets, by Diaz et al [SDLdSS96]. These are Petri nets with interval time constraints where nine different synchronization modes can be associated with delayable transitions. It can be shown that the guards corresponding to the different synchronization modes can be expressed compositionally as modal formulas in terms of the guards of the components.

We are currently studying the application of the results to define the semantics of the language used in the MADEUS tool for the specification of multimedia documents [JLSIR97]. This language allows the description of timing constraints by means of logical and relational operators used to express causality and synchronization relations. The interesting fact is that very often a combination of the three synchronization types is necessary to specify coordination. The results of the study will be published in [BST97].

5 Discussion

We present a general framework for the composition of hybrid automata. We show that from elementary hybrid actions, choice and parallel composition, complex systems can be defined.

The main difference with other approaches is that we associate with actions time progress conditions which specify for how long an enabled action may wait. Time progress conditions at a given state depend on the urgency of the enabled actions.

The big variety of choice and parallel composition operators results from the different ways enabledness and urgency of components can be combined. Contrary to untimed systems, it is necessary to use modalities to express different kinds of composition that are of practical interest. However, for many tractable subclasses of hybrid automata modal operators can be eliminated, e.g. for linear hybrid automata ([ACH⁺95]). In that case, modalities are used just for notation convenience and do not modify the basic model.

Different choice operators can be expressed in terms of a basic non-deterministic choice operator which combines the behaviors of the contributing actions so as to obtain maximum urgency. Restricting guards to respect priorities leads to the definition of less prompt choice operators. Other kinds of restrictions remain to be investigated.

Priority choice plays an important role for the definition of a parallel composition operator that respects maximal progress and avoids deadlock by means of appropriate interleaving actions.

The proposed framework is very general. Validation by practice is necessary. It is important to notice that so far AND-synchronization has been used for timed process algebras and the different timed extensions of the language Lotos [LL95] as well as for timed and hybrid automata. MAX-synchronization is implicitly used in the different extensions of timed Petri nets.

We believe that AND-synchronization is more appropriate for responsive synchronization, where process coordination is supposed to be strong enough to impose that all the timing constraints of the contributing actions are respected. This is often the case for input/output, sender/receiver synchronization where one of the actions is not submitted to deadline constraints. For example, in the train-gate example often mentioned in the literature [ACH⁺95] communication between the two processes (train and gate) is responsive as the gate reacts to input signals sent by the train. Applying AND-synchronization to obtain the product automaton means that the deadlines and upper bounds of each process must be respected. On the contrary, synchronization between the gate process and a car stopped before the gate should allow for waiting and MAX-synchronization seems more appropriate in this case. We believe that MAX-synchronization should be used to extend parallel composition of asynchronous processes à la CSP. When a hybrid system is obtained as the hybrid extension of an untimed system of communicating automata, it seems natural to use MAX-synchronization for actions that can wait indefinitely before synchronizing.

Finally, MIN-synchronization corresponds to a kind of (symmetric) interrupt and one can hardly imagine examples where the use of this synchronization mode alone suffices.

Acknowledgement : We thank S. Graf, S. Tripakis, E. Olive as well as M. Jourdan of the Opera project of INRIA for fruitful discussions about possible applications.

References

- [ACH⁺95] R. Alur, C. Courcoubetis, N. Halbwachs, T. Henzinger, P. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.
- [BK85] J. A. Bergstra and J. W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37(1):77–121, May 1985. Fundamental studies.
- [BS97a] S. Bornot and J. Sifakis. *On the composition of hybrid systems (complete version)*. 1997.
- [BS97b] S. Bornot and J. Sifakis. Relating time progress and deadlines in hybrid systems. In *International Workshop, HART'97*, pages 286–300, Grenoble, France, March 1997. Lecture Notes in Computer Science 1201, Springer-Verlag.
- [BST97] S. Bornot, J. Sifakis, and S. Tripakis. *Modeling Urgency in Timed Systems*. To appear in COMPOS'97, LNCS, September 1997.
- [JLSIR97] M. Jourdan, N. Layaida, L. Sabry-Ismail, and C. Roisin. An integrated authoring and presentation environment for interactive multimedia documents. In *4th Conference on Multimedia Modeling*, Singapore, November 1997. World Scientific Publishing.
- [JM94] M. Jourdan and F. Maraninchi. Studying synchronous communication mechanism by abstractions. In *IFIP Working Conference on Programming Concepts, Methods and Calculi*, San Miniato, Italy, June 1994. Elsevier Science Publishers.
- [KMP96] Y. Kesten, Z. Manna, and A. Pnueli. Verifying clocked transition systems. In *School on Embedded Systems*, Veldhoven, The Netherlands, November 1996.
- [LL95] G. Leduc L. Léonard. An extended lotos for the design of real-time systems. In *workshop DARTS'95*, Bruxelles, Belgium, November 1995.
- [SDLdSS96] P. Sènac, M. Diaz, A. Léger, and P. de Saqui-Sannes. Modeling logical and temporal synchronization in hypermedia systems. In *Journal on Selected Areas in Communications*, volume 14. IEEE, jan. 1996.
- [SY96] J. Sifakis and S. Yovine. Compositional specification of timed systems. In *13th Annual Symposium on Theoretical Aspects of Computer Science, STACS'96*, pages 347–359, Grenoble, France, February 1996. Lecture Notes in Computer Science 1046, Springer-Verlag.

An Equivalence Between a Control Network and a Switched Hybrid System

Linda Bushnell*, Octavian Beldiman ** and Gregory Walsh ***

Abstract. A simple model for ideal control networks is proposed in this paper. A model for hybrid systems due to Witsenhausen is extended by adding both a discrete output and input. This extended model is used for modeling an ideal network of interactive hybrid systems. An equivalence is established between the network model and the Witsenhausen model. This equivalence allows for simulating complicated systems, and extending different properties of Witsenhausen type systems to control network systems. A simple HVAC application is modeled using the above equivalence.

1 Introduction

Recently, the area of control networks has attracted increased interest. Control networks are seen as a possible way to analyze and design complex dynamical systems that either are scattered over a large area, or have real-time requirements that make the data transmission process a critical one. Several architecture standards have been developed in industry. Some of the most representative are the BACnet (building automation and control network) standard and the CAN (controller area network) standard.

The BACnet has been designed to provide a standard communication and environmental control network for commercial and government buildings and campus environments. The primary application for this standard is HVAC control. CAN has been designed primarily for automotive applications. For instance, Bräuninger *et. al.* [15] use the CAN standard to accommodate the growing need for data communications in trucks and busses. Several other standards are also available. Özgüner *et. al.* [14] use the control network standard FOSU (Ford-Ohio State University) to control automotive suspension.

Modeling and analysis of these networks have just started to develop. Although some introductory papers have been published ([2],[3]) very few papers discuss modeling or analysis issues. In [4] Walsh studies the race condition behavior for networks of hybrid systems. In [5], Tindell *et. al.* give bounds on the

** Department of Electrical and Computer Engineering, Box 90291, Duke University, Durham, NC 27708-0291, lgb@ee.duke.edu and ovb@ee.duke.edu

*** Department of Mechanical Engineering, University of Maryland, College Park, MD 20742, gwalsh@eng.umd.edu

* Dr. Bushnell is also with the U.S. Army Research Office, P.O. Box 12211, RTP, NC 27709-2211. This research was supported in part by the Army Research Office grant number DAAH04-93-D-0002.

message response times in a CAN network. More recently, Wong and Brockett [10] study the effect of the communication bandwidth constraints upon these systems.

The subject of control networks is strongly connected to the modeling and analysis of hybrid systems. A vast amount of literature can be found in that direction, in both the field of control and computer science ([6], [7], [8], [9], [11], [12]).

The Witsenhausen model is an older and simpler model [1], and seems to be a good starting point for modeling control networks. Our proposed model allows for distinguishing between the low-level, continuous dynamics and the high-level, discrete switching commands in the network. The high-level strategy is implemented with regard to the way the systems respond to events. We will see that this is actually established by choosing the discrete input transition sets and network priorities assignments.

A slightly modified version of the Witsenhausen model is presented in section 2. The modification consists of adding a discrete output to announce a transition of the discrete state. Then extensions of this model are presented in section 3. A network of these systems is proven to be equivalent to a Witsenhausen model in section 4. Each of the sections contains an example for the models presented. In section 5 we present a simple HVAC application and we used the equivalence in section 4 to build a simulator for it.

2 The Witsenhausen Model

Without loss of generality we consider only autonomous models. Time-varying vector fields may be made autonomous by adding time as a new state.

The Witsenhausen model for hybrid systems is developed as follows:

- state: $(m, x) \in M \times \mathbb{R}^n$, where M is a finite set of integers.
- transition sets: A transition from discrete state i to discrete state j is triggered when the continuous state x reaches a given set J_{ij} in \mathbb{R}^n . Define the arrival and departure sets as: $J_i^+ = \cup_j J_{ji}$ and $J_i^- = \cup_j J_{ij}$.

There are three assumptions on the transition sets:

Assumption 1 (1) for any three distinct indices i, j, k the sets J_{ij} and J_{ik} are distinct; (2) for all i in M , the set J_i^- is closed; and (3) for all i in M the sets J_i^- and J_i^+ are disjoint.

- vector field: $f : M \times \mathbb{R}^n \rightarrow \mathbb{R}^n$.
- trajectory: Suppose the initial state is m_0 . As long as $x \notin J_{m_0}^-$ the system evolves with the vector field $f(m_0, \cdot)$ as

$$\dot{x}(t) = f(m_0, x(t)).$$

When the state reaches the set $J_{m_0}^-$, because this set is closed and the path is compact, there is an earlier intersection time t_1 corresponding to $x_1 \in J_{m_0}^-$.

This x_1 can belong to only one of the sets $J_{m_0 i}$. Let it be $J_{m_0 m_1}$. From t_1 the discrete state is changed to m_1 , and the system evolves as:

$$\dot{x}(t) = f(m_1, x(t))$$

with initial condition $x(t_1) = x_1$.

- discrete output: Any time when a switching takes place, a discrete output is generated as a function of the new discrete state: $o_1 = d(m_1)$ at time t_1 .

The model may also include a control input $u : M \times \mathbb{R} \rightarrow \mathbb{R}^r$ with $u(m, \cdot)$ continuous. The vector space would then be defined as:

$$f : M \times \mathbb{R}^n \times \mathbb{R}^r \rightarrow \mathbb{R}^n.$$

The control is usually a function of time and the current state. The closed loop system, where the control is replaced by its dependence on the state is of the form:

$$\dot{x}(t) = f(m, x(t), u_m(t, x(t))) = g(m, x(t)), m \in M.$$

An example of a trajectory for a Witsenhausen system showing a transition from discrete state m_0 to discrete state m_1 is presented in figure 1.

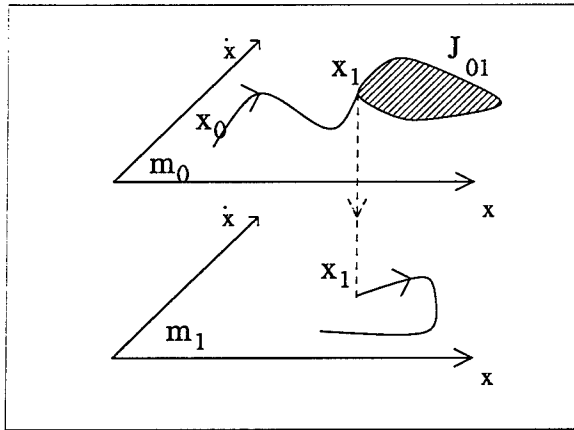


Fig. 1. An example of a trajectory for a Witsenhausen system.

To summarize, a Witsenhausen system is defined by the structure:

$(M, \Omega, f, d, \omega, \mathcal{J})$, where: M is a finite set of integers representing the discrete state space, Ω is a finite set of integers representing the discrete output set, $f : M \times \mathbb{R}^n \times \mathbb{R}^r$ is a function of class C^1 in the second argument, $d : M \times \mathcal{J} \rightarrow M$ is the discrete state transition function that indicates the next discrete state when the continuous state has reached one of the transition sets, $\omega : M \rightarrow \Omega$ is the discrete output function and $\mathcal{J} \in \mathcal{P}(\mathbb{R}^n)$ is the set of the transitions sets.

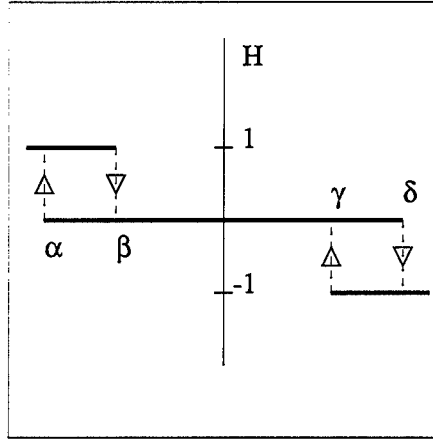


Fig. 2. Hysteresis function for Example 1.

To illustrate the above discussion, an example of a hybrid system due to Tavernini [9] is described as a Witsenhausen system:

Example 1. Consider the system:

$$\begin{aligned}\dot{x}_1 &= x_2 - \phi(x_1) \\ \dot{x}_2 &= H(\psi(x_1, x_2)) - \phi(x_2)\end{aligned}\quad (1)$$

where ψ , and ϕ are continuous, and H is a multi-valued function as in figure 2.

For this system we have three discrete states corresponding to the three possible values of H : $M = \{0, 1, 2\}$. For each discrete state we associate a discrete output by: $\omega : M \rightarrow \Omega$, $\omega(m) = m$, so $\Omega = \{0, 1, 2\}$. The field f is obtained for each discrete state replacing the corresponding value of H .

The set of transition sets is

$\mathcal{J} = \{J_{01}, J_{02}, J_{10}, J_{20}\}$, where: $J_{01} = \{(x, y) : \psi(x, y) \leq \alpha\}$, $J_{02} = \{(x, y) : \psi(x, y) \geq \delta\}$, $J_{10} = \{(x, y) : \psi(x, y) \geq \beta\}$ and $J_{20} = \{(x, y) : \psi(x, y) \leq \gamma\}$.

The discrete state transition function d is given by:

d	0	1	2
J_{01}	1	1	2
J_{02}	2	1	2
J_{10}	0	0	2
J_{20}	0	1	0

For example, if the system is in the discrete state 0, and the continuous state reaches the transition set J_{01} , then the discrete state will switch to state 1. If the continuous state then reaches the set J_{02} , the discrete state remains 1.

3 Extended Models

The Witsenhausen model covers a large class of hybrid systems, although some discrete phenomena (like autonomous or controlled jumps in the continuous state) are not taken into account. The model lacks the ability to exchange information with the external world. Even with adding a discrete output function, as in section 2 the model still has no discrete input capabilities.

In a network environment, however, the systems have to exchange information. To use a Witsenhausen model for these systems, it has to be augmented with discrete inputs. The effect of these discrete inputs will be to change the control law, and hence the vector field of the closed loop system, rather than the vector field of the open loop system. This extension is presented in this section.

3.1 State Dependent Control Switching

The first extension is the case where the control can be switched when the continuous state meets certain conditions, even if the vector field is not modified.

We assume that the control changes when the continuous state reaches some transition sets that respect condition similar to those from the Witsenhausen model.

For each discrete state the control may be one of a finite numbers of controls, depending on the continuous state. Then we can replace that discrete state with a finite number of discrete states, one for each possible control.

Using this method we obtain a new set of discrete states \tilde{M} . The only new switchings are those introduced by a change in control and we assumed that the transition sets for these switching respect Assumption 1.

For each discrete state $m \in \tilde{M}$, we have a continuous input:

$$u_m(t) = u_m(t, x(t)), m \in \tilde{M}.$$

The switching in the discrete state is governed by the continuous state, therefore this extended system is of Witsenhausen type.

Such systems are met when the control strategy is different for different regions in the state space.

Example 2. An example of such system is an inverted pendulum. For small deviation, the linearization is a good approximation and a classical controller, e.g., state feedback, can be used. When the pendulum is outside this region the controller can be switched to some other controller, e.g., a bang-bang controller.

3.2 Discrete External Commands

In a more general context, the control can be switched not only as a consequence of reaching some regions in the state space, but when receiving some external commands as well. A typical example of such switching is when a hybrid system receives a reset command: the control is switched to the control law corresponding to a given initial discrete state.

These systems are no longer equivalent to the Witsenhausen systems described in section 2. For this extension, the discrete commands can come at any time, no matter where the continuous state is. Therefore, the switching in the control, and thus the discrete state is not determined by the continuous state alone.

We will assume that at a given time the system receives a *finite* sequence of inputs. The equation for the continuous state is the same:

$$\dot{x}(t) = f(m, x(t), u(m, t))$$

but the switching in the state is no longer triggered only when $x(t)$ reaches J_m^- .

The system now has an additional discrete input $v_k \in V$, where V is a finite set. For any discrete states $m, n \in M$ there is a set $V_{mn} \subseteq V$, maybe empty, having the property that if the system was in the discrete state m and received a discrete input $v_k \in V_{mn}$, then the system switches to the discrete state n .

Denote by $V_m^+ = \bigcup_i V_{im}$ the arrival input set for state m , i.e., the set of discrete inputs that would switch the system to state m . Let $V_m^- = \bigcup_i V_{mi}$ be the departure input set for state m , that is, the set of all discrete inputs that would force the system to leave the discrete state m .

Assumptions similar to those for the transition sets in section 2 are needed:

Assumption 2 (1) for all $i, j, m \in M$ the sets V_{mi} and V_{mj} are distinct if $i \neq j$, and (2) for all $m \in M$ the sets V_m^- and V_m^+ are disjoint.

We can define $\mathcal{V} \in \mathcal{P}(V)$ the set of the input transition sets, and the discrete-input discrete-state transition function $v : M \times \mathcal{V} \rightarrow M$, such that $v(m, V_{mn}) = n$ (V_{mn} is the set of discrete inputs that would produce a transition from discrete state m in discrete state n).

After receiving a discrete input the system jumps in the corresponding state (which may be the same state). If the continuous state is in one of the transition sets for that state, the system jumps instantaneously in the new state.

In order to avoid a loop when connecting these system in a network we assume that the discrete output is triggered by the continuous state only (more precisely when the continuous state reaches one of the transition sets).

The mathematical model of these systems is therefore represented as

$(M, V, \Omega, f, d, \omega, \mathcal{J}, \mathcal{V}, v)$, where: $(M, \Omega, f, d, \omega, \mathcal{J})$ is a Witsenhausen system defined in section 2, V is a finite set of integers representing the discrete input set, $\mathcal{V} \in \mathcal{P}(V)$ is the set of discrete transition sets and $v : M \times \mathcal{V} \rightarrow M$ is the discrete-input discrete-state transition function that indicates the next discrete state corresponding to a given input

We will call such a system an *extended Witsenhausen system*.

Remark. The only assumption we made about the time distribution of the discrete input is that at a given time the system receives a finite sequence of inputs. Suppose that at a given time the system receives the inputs $\{v_1, v_2, \dots, v_k\}$. Then the resulting discrete state is obtained by sequentially processing these inputs. More exactly, the next discrete state will be $v(v(\dots v(m_0, v_1), \dots, v_{n-1}), v_n)$, where m_0 is the initial discrete state.

The extended Witsenhausen systems interact with the environment via discrete inputs and discrete outputs. A block diagram for such a system is shown in figure 3.

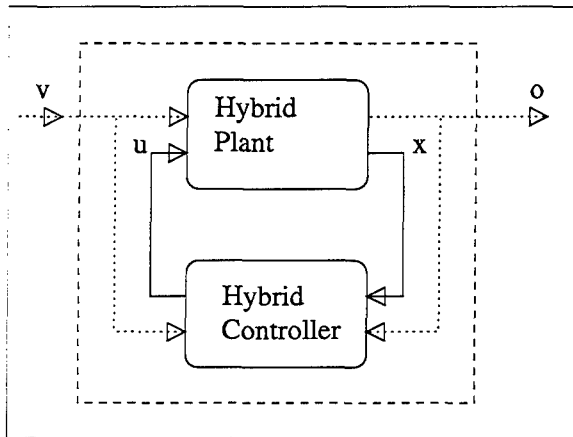


Fig. 3. Block diagram for an extended Witsenhausen system. The solid lines represent continuous signals and the dotted lines represent discrete signals.

These are the types of systems that one would expect in a network environment. Different systems in the network interact with each other, possibly changing their internal state when receiving messages from the other systems in the network. Messages are external commands that do not depend on the continuous state of the system, at least not directly. From this point of view, each individual system in the network is of the extended Witsenhausen type.

In the next section we will assume that all the extended Witsenhausen systems have the same input and output alphabet.

4 A Network of Systems

A control network is a collection of hybrid systems exchanging information to achieve a common goal. The term information stresses the fact that the systems exchange messages in response to some change in their discrete states. A control network is not the communication channel between a plant and its controller.

In this section, we will show a way to model a network composed of extended Witsenhausen systems.

There are a number of ways one can connect some extended Witsenhausen systems:

1. *parallel connection*: The systems have the same discrete input. This connection is trivial in the sense that each system evolves individually, without interacting with the others.

2. *serial connection*: The discrete output of a system is the discrete input of the next system. Such connection could model a pipelined environment, such as a manufacturing line. Each stage signals when it finishes processing an item. The the next stage receives this signal and knows that it has to start working on the item.
3. *loop connection*: The discrete output of one system is the discrete input of the second, and the discrete output of the second is the discrete input of the first. This can model any kind of hybrid plant / hybrid controller system.
4. *network connection*: This is the connection we are interested in for this paper. The discrete output of a system is the discrete input of all the other systems (except itself). Each of the systems has a priority, such that if several systems send a discrete output at the same time, they will be arranged in the order of the priorities. In that case all the systems will receive a finite ordered sequence of inputs, and will process it according with the above remark.

Note that the network connection described above neglects the transmission delays associated with the network. Hence, any two systems in the network can communicate with each other without transmission delays. A block diagram of a network connection of extended Witsenhausen systems is shown in figure 4.

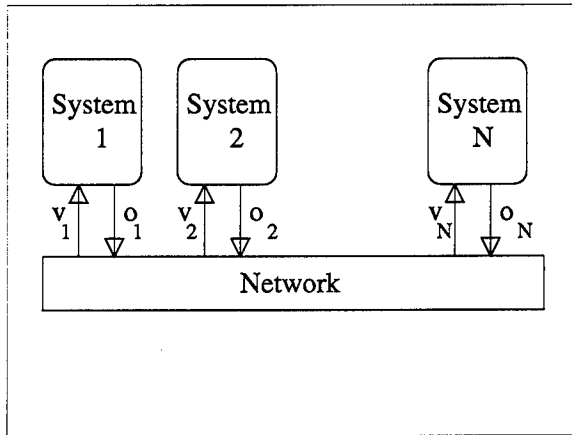


Fig. 4. Block diagram of a network connection.

Consider N extended Witsenhausen hybrid systems connected in a network, labeled in the decreasing order of their collision priorities:

$$(M_i, V_i, \Omega_i, f_i, \omega_i, d_i, \mathcal{J}_i, \mathcal{V}_i, v_i), i = 1, \dots, N.$$

We now state the main proposition of this paper:

Proposition 1. *A network of N extended Witsenhausen hybrid systems is equivalent to a Witsenhausen system obtained by concatenating the individual systems.*

More precisely, the system obtained by concatenating the individual systems is $(M, \Omega, f, d, \omega, \mathcal{J})$, where:

- $M = M_1 \times M_2 \times \dots \times M_N$
- $\Omega = \Omega_1 \times \Omega_2 \times \dots \times \Omega_N$
- $f = [f_1, f_2, \dots, f_N]^T$
- $\omega = (\omega_1, \omega_2, \dots, \omega_N)$
- $\mathcal{J} = \{A_1 \times A_2 \times \dots \times A_n \mid A_i \in \mathcal{J}_i \text{ or } \exists B_i \in \mathcal{J}_i, A_i = \mathcal{C}(B_i) \forall i \text{ and there is at least an } i \text{ for which } A_i \in \mathcal{J}_i\}$, where \mathcal{C} is the usual notation for set complement.
- d is the discrete state transition function which is defined as follows:
 Let $m = (m_1, m_2, \dots, m_N) \in M$ and $J = (A^1, A^2, \dots, A^N) \in \mathcal{J}$. We have to define $n = (n_1, n_2, \dots, n_N)$ such that $n = d(m, J)$.
 Construct the sequence $\{i_k\}_{1 \leq k \leq r}$ of indices for which $A_{i_k} \in \mathcal{J}_{i_k}$ (in increasing order). Then we can also construct the sequence $\{v_k\}_{1 \leq k \leq r}$ such that $v_k = \omega_{i_k}(m_{i_k}, A_{i_k})$.
 Take $n_{i_k} = d_{i_k}(m_{i_k}, A_{i_k})$.
 For the other components, construct iteratively the next state, considering as a sequence of discrete inputs the sequence $\{v_k\}$.
 Note that the case when the sequence $\{i_k\}$ has more than one element corresponds to a collision (because several systems will try to send their output on the network at the same time). Then the network will broadcast their output in the order of their priority (due to our assumption that the systems has been labeled in decreasing order of their priorities, e.g., system 1 has the highest priority).

Remark. For this composed system the continuous state is the concatenation of all individual continuous states:

$$x(t) = [x_1(t), x_2(t), \dots, x_n(t)]^T$$

and the discrete state takes values in the Cartesian product of the discrete state spaces of each system (still a finite set):

$$m = (m_1, m_2, \dots, m_n) \in M_1 \times M_2 \times \dots \times M_n.$$

Proof. We have to prove two things: first we have to prove that the transition sets for the network respect the three assumptions needed for the Witsenhausen model, and then we have to prove that the switching of the discrete state for the network is triggered by its continuous state.

We'll consider $N = 2$. The proof for a general N is a simple extension of this case.

Suppose we aggregate two Witsenhausen models. One of them has continuous state $x(t) \in M$ and discrete state $d \in D$, and in the other has continuous state $y(t) \in N$ and discrete state $e \in E$.

Denote by J_{xd}^+ and J_{xd}^- the arrival set into and the departure set from the discrete state d for the first system, and by J_{ye}^+ and J_{ye}^- the arrival set into and the departure set from the discrete state e for the second system.

Since the two system are Witsenhausen, the sets J_{xd}^- and J_{ye}^+ are closed (in M , respectively N).

Let's denote by J_{de}^+ and J_{de}^- the arrival set into and the departure set from the discrete state (d, e) for the composed system (the model for the network).

If we ignore the interaction between the two systems (the fact that the switching in the discrete state of one of the systems could generate a message through the network that may trigger the switching of the discrete state for the second system) then we have:

$$\begin{aligned} J_{de}^- &= (J_{xd}^- \times N) \cup (M \times J_{ye}^-) \\ J_{de}^+ &= (J_{xd}^+ \times \mathcal{C}(J_{ye}^-)) \cup (\mathcal{C}(J_{xd}^-) \times J_{ye}^+) \end{aligned}$$

Indeed, the network leaves state (d, e) either when the first system leaves state d or when the second system leaves state e . Similarly, the network comes into state (d, e) either when the first system comes into state d and the second does not leave state e or the other way around.

These new sets respect the Assumption 1:

1. To prove that $J_{de}^- \cap J_{de}^+ = \emptyset$, suppose that there is a point in this intersection: $(x, y) \in J_{de}^- \cap J_{de}^+$. Then $(x, y) \in J_{de}^-$. So either $x \in J_{xd}^-$ or $y \in J_{ye}^-$. In both cases (x, y) cannot belong to J_{de}^+ , thus the departure and arrival sets are disjoint.
2. J_{de}^- is closed because it is a finite union of closed sets.
3. The third assumption (the fact that the sets of the form $J_{(de)(fg)}$ are disjoint) is ensured by the fact that the sets $J_{x(d)(f)}$ and $J_{x(e)(g)}$ are disjoint since they are transition sets for Witsenhausen systems (and the former sets are simple Cartesian products of the later sets).

However this does not take into account the switching generated by sending messages on the net. Assume now that there is some interaction between the two systems. This does not change the departure sets (only their distribution), but affects the arrival sets. The effect is that one arbitrary transition set, let's say $J_{(mn)(dn)}$ is added to J_{de}^+ (the transition of m to d could send a message to trigger n to e). Then we have a problem if $J_{(mn)(dn)} \in M \times J_{xe}^-$.

But the way we defined the transition function for the composed system allows us to get rid of this problem: the discrete state of the network goes directly to the final state, avoiding the possible transparent state. So, even if we take into account the messaging, the transition sets still verify the assumptions from the Witsenhausen model.

The only thing left to prove is that a change in the discrete state is triggered only when the continuous state reaches one of the sets in \mathcal{J} .

A change in the discrete state of the overall system happens if and only if some of the individual systems change their discrete states. An individual system

p can change its discrete states i in two cases: either its continuous state reached the departure set J_i^- or an external command in the departure input set V_i^- has been received through the network.

In the first case the change in the discrete state is triggered by the continuous state. In the second case, let q be the system that issued the command. System q can send an output on the net only when its continuous state reaches one of the transition sets. So in this case, the change in the discrete state of the system p is triggered by the continuous state of the system q . That means that in general, the change in the discrete state of the composed system is triggered by its continuous state.

This proves the fact that the system obtained by concatenating all the individual system from the network is of Witsenhausen type.

□

Example 3. Consider two extended Witsenhausen systems:

$$\dot{x}_i(t) = f_i(q_i, x_i), i = 1, 2$$

having two discrete states $q_i \in \{0, 1\}$ with transition sets: $J_{01}^{(i)} = \{x | x \leq a_i\}$ and $J_{10}^{(i)} = \{x | x \geq b_i\}$, $i = 1, 2$ (the notation J_{mn} actually means that $d(m, J_{mn}) = n$).

Take $\Omega_1 = \Omega_2 = 0, 1$, $\omega_1(m) = \omega_2(m) = m$, and $V_1 = V_2 = 0, 1$.

Suppose $\mathcal{V}_i = \{V_{01}\} = \{\{1\}\}$ (if the system was in state 0 and receives an input 1 then it will switch to state 1). The equivalent system for a network composed of these two systems is presented as:

$$M = \{0, 1\} \times \{0, 1\} = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$$

$$\Omega = \Omega_1 \times \Omega_2 = \{(0, 0), (0, 1), (1, 0), (1, 1)\}$$

$$f = [f_1, f_2]^T$$

$$\omega : M \rightarrow \Omega, \omega(m) = m$$

$$\mathcal{J} = \{J_{01}^{(1)} \times J_{01}^{(2)}, J_{01}^{(1)} \times J_{10}^{(2)}, J_{01}^{(1)} \times \mathcal{C}(J_0^{(2)-}), J_{01}^{(1)} \times \mathcal{C}(J_1^{(2)-}), J_{10}^{(1)} \times J_{01}^{(2)}, J_{10}^{(1)} \times J_{10}^{(2)}, J_{10}^{(1)} \times \mathcal{C}(J_0^{(2)-}), J_{10}^{(1)} \times \mathcal{C}(J_1^{(2)-}), \mathcal{C}(J_0^{(1)-}) \times J_{01}^{(2)}, \mathcal{C}(J_0^{(1)-}) \times J_{10}^{(2)}, \mathcal{C}(J_1^{(1)-}) \times J_{01}^{(2)}, \mathcal{C}(J_1^{(1)-}) \times J_{10}^{(2)}\}$$

Denote by J_1, \dots, J_{12} the elements of \mathcal{J} , then: $d((0, 0), J_1) = d((0, 0), J_2) = d((0, 0), J_3) = (1, 1)$, $d((1, 0), J_4) = (0, 0)$, $d((1, 0), J_5) = d((1, 0), J_6) = (1, 1)$, $d((0, 1), J_7) = (0, 0)$, $d((0, 1), J_8) = d((0, 1), J_9) = (1, 1)$, $d((1, 1), J_{10}) = (1, 0)$, $d((1, 1), J_{11}) = (0, 1)$, and $d((1, 1), J_{12}) = (0, 0)$.

This example illustrates the equivalence between a control network of two extended Witsenhausen systems and a Witsenhausen system.

5 A Simple HVAC Application

In this section we present a very simple temperature control problem to illustrate the extended Witsenhausen model and the equivalent model of a control network.

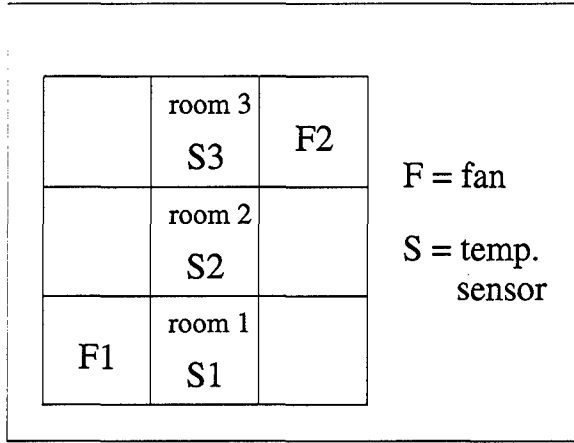


Fig. 5. Plan of a building.

In figure 5 we show a plan of a building with 9 rooms, 2 ventilation fans and 3 temperature sensors. We will consider that the sensors and the fans are connected by a network so that they can exchange information, and that there is no central controller involved.

We want to model this system so that we can simulate its behavior for different control strategy for the fans.

The fans can have only two states: on and off. We denote by m_1 the state of the first fan ($m_1 = 0$ if the first fan is off and $m_1 = 1$ if the first fan is on), and by m_2 the state of the second fan. If a fan is on, the adjacent rooms are heated with a rate r , and the third room is heated with a rate $r/2$.

We will assume very simple equations for the temperature in the rooms (u represents the temperature):

- room 1: $u'_1 = -u_1 + rm_1 + 0.5rm_2$
- room 2: $u'_2 = -u_2 + rm_1 + rm_2$
- room 3: $u'_3 = -u_3 + 0.5rm_1 + rm_2$

The sensors will send a signal if the temperature in that room is above an upper limit T_{max} or below a lower limit T_{min} . We notice that we don't need to model the fans because their influence is implicit in the temperature equations for the rooms. We will consider the following control strategy:

- if the temperature in room 1 is above T_{max} switch fan 1 off.
- if the temperature in room 1 is below T_{min} switch fan 1 on.
- if the temperature in room 3 is above T_{max} switch fan 2 off.
- if the temperature in room 3 is below T_{min} switch fan 2 on.
- if the temperature in room 2 is above T_{max} switch both the fans off.
- if the temperature in room 2 is below T_{min} switch both the fans on.

Each room can be modeled as an extended Witsenhausen model with four discrete states, each state corresponding to a possible combinations of states for the fans.

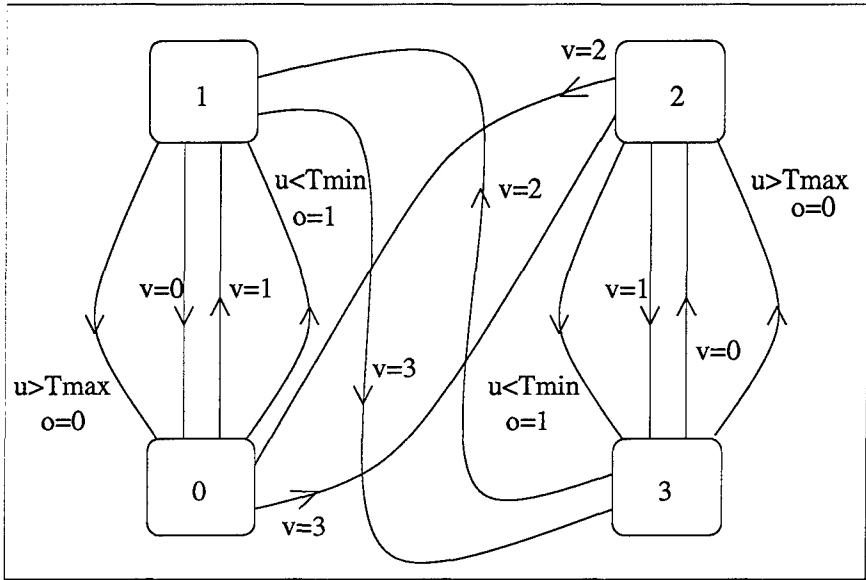


Fig. 6. Underlying finite state automaton for room 1.

The equations corresponding to room 1, for each discrete state are:

- $q_1 = 0 : u'_1 = -u_1$ (both fans are off)
- $q_1 = 1 : u'_1 = -u_1 + r$ (first fan on, the second off)
- $q_1 = 2 : u'_1 = -u_1 + 0.5r$ (first fan off, the second on)
- $q_1 = 3 : u'_1 = -u_1 + 1.5r$ (both fans on)

The equations for the other two rooms are similar.

Figure 6 presents the underlying finite state automaton for the first room. Note that transitions can occur either if the temperature passes the limits, or if a discrete input, v , is read from the network. This discrete input was generated by another room whose temperature passed the limits. Below each transition due to the continuous state, the discrete output, o , is specified.

The control strategy for the fans is coded in the way the discrete inputs/outputs are defined in the control network. We define six messages that can be exchanged between rooms, having the following meaning: $v = 0$: fan 1 is off, $v = 1$: fan 1 is on, $v = 2$: fan 2 is off, $v = 3$: fan 2 is on, $v = 4$: both fans are on and $v = 5$: both fans are off.

Once we have defined each extended Witsenhausen system that is connected to the network, we can form the equivalent Witsenhausen model for the network.

The continuous state will be $u = (u_1, u_2, u_3) \in \mathbb{R}^3$. The discrete state will be $q = (q_1, q_2, q_3)$. The transition sets are Cartesian products of the individual transition sets, or their complements.

For instance the transition set from state $(0, 0, 0)$ to state $(1, 1, 1)$ is:

$$(-\infty, T_{min}] \times (T_{min}, \infty) \times (T_{min}, \infty).$$

Using this model we simulated the network in Matlab, and the behavior of the system for two different fan control strategies is shown in figure 7. In the simulator used, the transitions sets were not precomputed, but once the system crossed one of the limit temperatures, the new discrete state was established.

In figure 7 the system is simulated for two fan control strategies: the first one is the one described above, and the second one is very similar, with the only difference that when the temperature in room 2 reaches one of the limits, only fan 1 is turned on or off.

Using the equivalence to a Witsenhausen model, from Proposition 1, a network of systems is very easy to simulate. Otherwise for each network one would have to write a specific simulator.

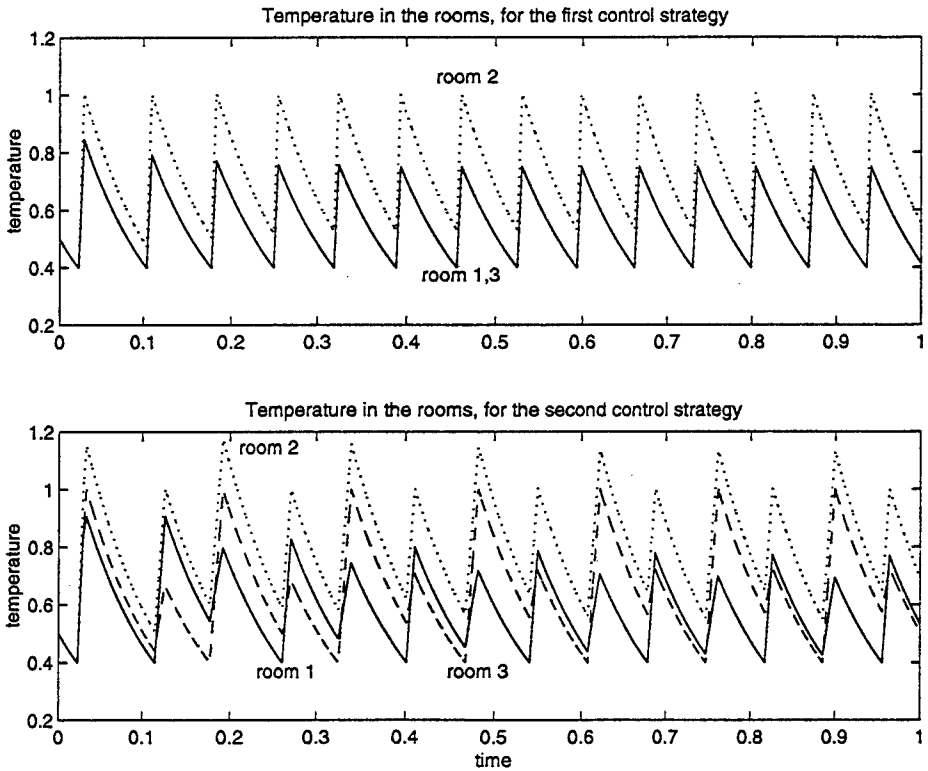


Fig. 7. Temperature in the three rooms for the two fan control strategies. We used $T_{max} = 1$, $T_{min} = 0.4$, $r = 1$.

6 Conclusions

The equivalence between a network of extended Witsenhausen systems and a Witsenhausen system has been proven in this paper. This equivalence offers us a way to analyze and design control networks. Once we know certain properties about Witsenhausen systems, we can apply them directly to the network. For instance, in [6], Branicky extends the Lyapunov stability theory to switched systems, which are similar to the Witsenhausen systems discussed in this paper. This extended theory can be a way to analyze the stability of the control network. Also, in [1] Witsenhausen proves some necessary conditions for optimal control of his models, which can be applied to the control network to solve optimal control problems.

The main assumption in our derivation was that the control network has no communication delays. In a real network this does not happen, although for control networks, which are designed for high speed small packages, the delays are very small. A critical problem occurs when several component systems change their states simultaneously. Then the sequence of messages will come scattered in time, requiring some kind of robustness for the system that has to process that information. More precisely the transition sets corresponding to two successive discrete sets should not be arbitrarily close. This problem is a future research topic.

Additional future research includes extending stability results to Witsenhausen systems, as well as extending the limit cycle theory of nonlinear systems. Also the effect of the assumptions made about the network, and ways to introduce perturbation factors that would model the environmental connection between different systems in the network will be studied.

References

1. H. S. Witsenhausen. "A class of hybrid-state continuous time dynamic systems," *IEEE Transactions on Automatic Control*, 11(2):161-167, v1966.
2. R. S. Raji. "Smart networks for control," *IEEE Spectrum*, June 1994, p. 49-55.
3. D. Radford. "Spread-spectrum data leap through ac power wiring," *IEEE Spectrum*, November 1996, p. 48-53.
4. G. Walsh. "On race conditions for networked control systems," in *Proceedings of the 30th CISS*, Princeton, NJ, March 1996, p 411-415.
5. K. Tindell, A. Burns and A.J. Wellings. "Calculating controller area network (CAN) message response times," *Control Eng. Practice*, vol. 3, no. 8, p. 1168-1169, 1995.
6. M. S. Branicky. "Studies in hybrid systems: modeling, analysis, and control," Ph.D. dissertation, MIT, June 1995.
7. A. Back, J. Guckenheimer and M. Myers. "A dynamical simulation facility for hybrid systems," in Grossman et al. [13], p. 255-267.
8. A. Nerode and W. Kohn. "Models for hybrid systems: Automata, topologies stability." In Grossman et al. [13], p. 317-356.

9. L. Tavernini. "Differential automata and their discrete simulators." in *Nonlinear analysis, Theory, Methods and Applications*, 11(6): 665-683, 1987.
10. W. S. Wong and R. W. Brockett. "Systems with finite communication bandwidth constraints - Part I: State estimation problems," in *IEEE Transactions on Automatic Control*, 42(9): 1294-1299.
11. R. W. Brockett. "Hybrid models for motion control systems." in H.L. Trentelman and J.C. Willems, editors, *Essays in Control: Perspectives in the Theory and its Applications*, p. 29-53, 1993.
12. R. W. Brockett. "Dynamical systems and their associated automata." in U.Helmke, R. Menniken and J.Saurer, editors, *Systems and networks: Mathematical theory and applications*. Akademie Verlag, Berlin 1994.
13. R. L. Grossman, A. Nerode, A. P. Ravn, H. Rischel editors *Hybrid systems*, volume 736 of *Lecture notes in computer science*. Springer-Verlag, New York, 1993.
14. U. Özgüner, H. Goktas, H. Chan. *Automotive suspension control through a computer communication network*. 1st IEEE Conference on Control Application, 1992.
15. J. Bräuninger, R. Emig, T. Küttner and A. Löffle. *Controller Area Network for Truck and Bus Application.*, SAE Transactions, v 99, sect 2, 1990, p 704-714.

Hybrid cc with Interval Constraints

Björn Carlson *

Vineet Gupta**

Abstract. Hybrid cc is a constraint programming language suitable for modeling, controlling and simulating hybrid systems, i.e. systems with continuous and discrete state changes. The language extends the concurrent constraint programming framework with default reasoning and combinators for programming continuous behavior. The most important constraint systems used in Hybrid cc are nonlinear equations and ordinary differential equations over intervals. We describe the implementation of the Hybrid cc interpreter and constraint solvers, and evaluate the performance using some example programs.

1 Introduction

Hybrid cc [GJS97, GJSB95] is a compositional, declarative language, based on constraint programming, which enables modeling and simulation of hybrid systems in one framework. In Hybrid cc, a hybrid system is specified by a set of constraints on its temporal behavior. Each constraint describes an internal relationship of the system, e.g. the heat loss of a container as a function of time, or the acceleration as it depends on mass. The constraints are based on standard formalisms used in physics and engineering, such as differential equations and algebraic equations. Discrete events and state changes, such as turning on a heater when the ambient temperature drops too low, are specified using the combinators of concurrent constraint programming [Sar93] and default logic [Rei80]. The formal operational semantics of Hybrid cc is described in [GJS97].

This paper presents an implementation of Hybrid cc. We have chosen an interval constraint system for our implementation, since this gives us the ability to model some uncertainty in the parameters. The two most important classes of constraints used in our implementation are (nonlinear) algebraic and ordinary differential equations. Algebraic constraints are solved by interval propagation using indexicals, interval splitting, the Newton-Raphson method and the Simplex algorithm. Differential equations are integrated using a version of the fourth-order Runge-Kutta method with adaptive stepsize, modified for interval variables. We use constraint propagation to solve the simultaneous differential equations.

Interval constraints provide Hybrid cc with the expressive power required for many modeling problems [GSS95], where inequalities are used to express physical

* Netscape Communications, 650 E. Middlefield Road, Bldg.1, Mountain View CA 94043; bjorn@netscape.com

** Caelum Research Corporation, NASA Ames Research Center, M/S 269-2, Bldg 269, Rm 127, Moffett Field CA 94035, vgupta@ptolemy.arc.nasa.gov

constraints like bounds on force magnitudes etc. In addition, many physical systems are imprecise in nature, i.e. we cannot construct a perfectly accurate model. The imprecision is captured by interval constraints. By using constraint propagation inside the numerical integrator we are further able to strengthen the precision of the integration by adding redundant constraints, which narrow the divergence (see example in Section 3) that follows from imprecise initial conditions.

Our implementation is based on an interpreter and compiler written in C and Yacc. The compiler translates each program into a graph of expressions, which is interpreted combinator by combinator. The interpreter keeps a constraint store similar to the store of traditional constraint programming languages. The memory is managed using a conservative garbage collector for C.

The implementation is easily embeddable in other systems. For example, we have integrated Hybrid cc with Java, both as a Win32 dynamic library with an API for compiling and running Hybrid cc under Windows 95 and Windows NT, and as a remote procedure call interface for compiling and running Hybrid cc remotely. We have also developed a modeling client in Java with support for visual Hybrid cc programming, and graphical output, both as graph plots and 2.5 D animations generated by sampling variables during the execution of an Hybrid cc program.

The performance of Hybrid cc on interval constraint benchmarks shows that its interval propagation is comparable with the best interval solvers *e.g.* clp(Newton) [VMK95]. Clearly, our interval version of the Runge-Kutta method is not as fast as standard libraries for integration over real-valued variables. However, by using interval variables and constraint propagation inside the numerical solver, we get a more flexible and robust system for modeling with differential equations.

The paper is structured as follows. In Section 2 we give a brief introduction to Hybrid cc and give its operational semantics. We then give a description of the constraint solvers in Section 3. Finally we conclude with a comparison with related work and an evaluation of the performance of the interpreter.

2 Hybrid cc – the language and its use

Hybrid cc extends concurrent constraint programming with defaults, continuous combinators and objects. The basic set of combinators in Hybrid cc are as follows:

c	tell the constraint c
if d then A	if d holds, reduce to A
unless d then A	reduce to A unless d holds
A, B	parallel composition
new V in A	V is local to A
forall $C(X)$ do A	do $A[I/X]$ for each instance I of class C
hence A	execute A at every instant after now
$X(T_1, \dots, T_k)$	execute X with parameters T_1, \dots, T_k

The constraint system we have implemented is as follows.

Continuous Constraints. These constraints assert equalities and inequalities over arithmetic terms. The syntax is as follows:

$$\begin{aligned}
 \text{ContConstr} &::= \text{Term RelOp Term} \mid \text{cont}(\text{LVariable}) \\
 \text{RelOp} &::= = \mid > \mid < \mid = \\
 \text{Term} &::= \text{LVarExpr} \mid \text{Constant} \mid \text{Term BinOp Term} \mid \text{UnOp}(\text{Term}) \\
 &\quad \mid \text{Term}' \\
 \text{LVarExpr} &::= \text{LVariable} \mid \text{UVariable.LVarExpr} \\
 \text{BinOp} &::= + \mid - \mid * \mid / \mid ^ \\
 \text{UnOp} &::= - \mid \sin \mid \cos \mid \log \mid \exp \mid \text{prev}
 \end{aligned}$$

LVariables are variable names which start with a lowercase character while *Constants* are floating point numbers. *LVarExprs* are *LVariables* or property expressions (see paragraph below). The semantics of most constructs is as expected. For example, $\exp(x)$ is the exponential function e^x , Term' denotes the derivative of Term with respect to the implicit variable time.

$\text{cont}(x)$ asserts that x is continuous. Thus, **always** $\text{cont}(x)$ asserts that x is always continuous. The effect of asserting $\text{cont}(x)$ in a point phase is that the value of x in the point phase is set to the value of x at the end of the previous interval phase. Note that asserting $x' = 3$ automatically asserts that x is continuous, as differentiability implies continuity.

Ask arithmetic constraints also allow the Relops $<$, $>$, $!$, $=$.

Non-arithmetic Constraints. These are constraints on non-arithmetic variables — these variables do not change their values continuously. The syntax for such constraints is given by

$$\begin{aligned}
 \text{DConstr} &::= \text{UVarExpr} \mid \text{UVarExpr} = \text{DE Expr} \\
 \text{UVarExpr} &::= \text{UVariable} \mid \text{UVarExpr.UVarExpr} \\
 \text{DE Expr} &::= \text{UVarExpr} \mid \text{String} \mid (\text{VarList})[\text{VarList}]\text{HccProg} \\
 &\quad \mid (\text{VarList})\text{HccProg} \mid \text{UVarExpr}(\text{VarList})[\text{VarList}]\text{HccProg} \\
 \text{VarList} &::= \text{UVariable} \mid \text{LVariable} \mid \text{VarList}, \text{VarList}
 \end{aligned}$$

A *UVariable* is a variable name starting with an uppercase character. *UVarExpr*'s are *UVariables* or property expressions (see below). *HccProg* is any Hybrid cc program, defined above.

A *DConstr* given as a *UVarExpr* is a signal constraint. These constraints are typically used to communicate to some other statement of code that a certain state is reached or a certain property is true.

A *String* is any string of characters enclosed within double quotes. These are mostly used for properties of objects — i.e. **Switch** = "on".

The constraint $X = (V_1, \dots, V_k)\text{HccProg}$ sets up a closure definition. It defines X to behave like $\lambda V_1 \dots \lambda V_k.\text{HccProg}$ with the exception that X can only be β -reduced when all k arguments are given. The factorial function can now be defined recursively as follows:

$$P = (n, m, Q) \{ \text{if } (n > 0) \text{ then new } x \text{ in } \{Q(n-1, x, Q), m = x * n\}, \\
 \text{if } (n = 0) \text{ then } m = 1 \}$$

so the call $P(n, x, P)$ computes $x = n!$. Closures are first class objects, and can be passed around as data.

The constraint $C = B(V_1, \dots, V_k)[P_1, \dots, P_l]A$ sets up a class definition. It defines a class C , where the constructor takes k arguments, and the properties of C are named P_i , $1 \leq i \leq l$. Note that a property P_i can point to a closure, this is how methods are defined. A property is treated as a variable inside A . The functor B is optional, and if used, it must be constrained to a (base) class from which C inherits all the properties. The code in A is used for defining any instance of C (see below).

Objects are created using the same syntax as for a closure call $C(Name, t_1, \dots, t_k)$ where C will be bound to a class definition. The argument $Name$ is mapped to a property named *Self*. A property x can be referred inside the code as x , but from outside it must be referred as $Name.x$. Any code in C is run with $Self = Name$ to initialize the object.

Ask constraints for the above have a similar syntax, except that the $Relop \neq$ is also allowed. Note that asks do not make sense for closure and class definitions, as we do not perform any unification on these. Thus asking $A = (M)HccProg$ will always answer unknown.

Computational model of Hybrid cc. The computational model is based on reductions of statements. Let σ denote a variable store, i.e. a set of interval constraints $x \in [a, b]$, string, atom, closure and class constraints. An Hybrid cc system consists of a store, a set of Hybrid cc statements, and some auxiliary structures.

An Hybrid cc system alternates between being in a *point phase* and in an *interval phase*. The initial phase is a point. Let A be the initial statement to be reduced. By the semantics of the operators, defined below, a stable point is eventually reached for A (we assume throughout that no infinite sequence of reductions occurs), where all constraints have been propagated, and all reductions of statements in A have been completed.

Now, either the stable store σ is inconsistent, and the computation is aborted, or it is consistent. In the latter case, the computation enters the interval phase. The statements to be reduced in this phase consist of each B that was reduced by **hence** B in the point phase, together with the statement **hence** B itself (remember that **hence** B means that B is to be reduced continuously and forever).

Similar to the point phase, all the statements in the set described above are reduced until a stable point is reached. This determines the set of constraints that are continuously true in the current phase, and the set of statements to be reduced at the next point phase. The length of the interval phase is the longest interval during which the constraint set is unchanged — this is ensured by making sure that none of the asked constraints changes status. For example, consider the program

$$x = 0, \text{ hence } \{x' = 1, \text{ if } (x = 2) \text{ then } y = 1\} \quad (1)$$

In the interval phase following $x = 0$, x evolves continuously according to $x' = 1$, through the interval $(0, 2)$ until $x = 2$ is about to become true. At this point the set of constraints may change, so the next point phase is started.

We first describe the reduction rules for each operator of Hybrid cc, and then provide the algorithm for the interpreter. The following reduction rules apply in either phase. Γ denotes a set of Hybrid cc program fragments, σ denotes the store, **next** the set of program fragments to be run in the next phase, and **default** a set of suspended else statements. The expression $\sigma \vdash c$ denotes entailment checking.

$$\begin{array}{ll}
\text{Tell} & \langle (\Gamma, c), \sigma, \text{next}, \text{default} \rangle \rightarrow \langle \Gamma, \sigma \cup \{c\}, \text{next}, \text{default} \rangle \\
\text{Ask} & \frac{\sigma \vdash d}{\langle (\Gamma, \text{if } d \text{ then } A), \sigma, \text{next}, \text{default} \rangle \rightarrow \langle (\Gamma, A), \sigma, \text{next}, \text{default} \rangle} \\
\text{Unless} & \langle (\Gamma, \text{unless } d \text{ then } A), \sigma, \text{next}, \text{default} \rangle \rightarrow \\
& \quad \langle \Gamma, \sigma, \text{next}, (\text{default}, \text{unless } d \text{ then } A) \rangle \\
\text{Par} & \langle (\Gamma, (A, B)), \sigma, \text{next}, \text{default} \rangle \rightarrow \langle (\Gamma, A, B), \sigma, \text{next}, \text{default} \rangle \\
\text{Forall} & \frac{\sigma \vdash I_1, \dots, I_n \text{ are instances of } C}{\langle (\Gamma, \text{forall } C(X) \text{ do } A), \sigma, \text{next}, \text{default} \rangle \rightarrow} \\
& \quad \langle (\Gamma, A[I_1/X], \dots, A[I_n/X]), \sigma, \text{next}, \text{default} \rangle \\
\text{New} & \langle (\Gamma, \text{new } X \text{ in } A), \sigma, \text{next}, \text{default} \rangle \rightarrow \langle (\Gamma, A[Y/X], \sigma, \text{next}, \text{default}) \\
& \quad (Y \text{ new}) \\
\text{Call} & \frac{\sigma \vdash P = (V_1, \dots, V_k)A}{\langle (\Gamma, P(t_1, \dots, t_k)), \sigma, \text{next}, \text{default} \rangle \rightarrow} \\
& \quad \langle (\Gamma, A[t_1/V_1, \dots, t_k/V_k]), \sigma, \text{next}, \text{default} \rangle
\end{array}$$

The rule for **hence** A differs in point and interval phases.

$$\begin{array}{ll}
\text{Hence Point} & \langle (\Gamma, \text{hence } A), \sigma, \text{next}, \text{default} \rangle \rightarrow \\
& \quad \langle \Gamma, \sigma, (\text{next}, \text{hence } A), \text{default} \rangle \\
\text{Hence Interval} & \langle (\Gamma, \text{hence } A), \sigma, \text{next}, \text{default} \rangle \rightarrow \\
& \quad \langle (\Gamma, A), \sigma, (\text{next}, A, \text{hence } A), \text{default} \rangle
\end{array}$$

The **tell** rule propagates the effects of the constraints using the algorithms described in the next section. The combinator **forall** also suspends such that if any further instance I of C is created in the current phase, the combinator adds $A[I/X]$ to Γ . Similarly, for $\sigma \vdash c$, if c is neither detected true or false in the current store, the statement that contains c is suspended and reconsidered whenever any of the variables in c changes value (e.g. is pruned).

The algorithm for the interpreter is the same in both phases, except for the integration at the end of the interval phase. It involves the following steps:

1. Run the reduction rules on the current $\langle \Gamma, \sigma, \text{next}, \text{default} \rangle$, till no further reductions can take place.
2. If σ is inconsistent, return 0.
3. If **default** is empty, return 1.

4. Remove one statement from **default**— **unless** c **then** A . If $\sigma \vdash c$, go to step 3.
5. Add A to Γ . Run the interpreter on the current state. If the result is 1 and $\sigma \not\vdash c$, return 1.
6. Undo the effects of the previous step by backtracking. Run the interpreter on the current state. If the result is 1 and $\sigma \vdash c$, return 1. Otherwise return 0.

Note that the effect of the last three steps is that a maximal set of defaults is chosen and executed (similar to the maximal extensions of [Rei80]). This is similar to the causal loops in synchronous languages [Hal93, BB91, Har87, SJG96]. There can be many different maximal sets, our interpreter chooses any one randomly. For example **unless** X **then** Y , **unless** Y **then** X can reduce to either X or Y but not both. If no maximal set exists, as for the statement **unless** X **then** X , then the computation must be aborted.

A Hybrid cc program A is run as follows.

1. Run interpreter with $\Gamma = A$, and empty σ , **next** and **default** in the point phase. If the result is 0, abort.
2. Run the interpreter in the interval phase with $\Gamma = \mathbf{next}$, as returned by the point phase. σ , **next** and **default** are again empty. If the result is 0, abort. Record all the tells, and also the ask constraints that were checked during the phase.
3. Integrate the arithmetic constraints that were told in the previous step, until one of the ask constraints changes status (i.e. goes from false to true or unknown, etc.). Go to step 1 with $\Gamma = \mathbf{next}$.

Hybrid cc also contains various constructs from synchronous programming, e.g. **do** A **watching** c , **when** c **do** A , but since their behavior can be derived from the behavior of the above constructs we omit them here.

Implementation. Our interpreter implements essentially the above algorithm, with a few changes. For example, the interpreter is not recursive, but uses stacks for managing the backtracking. The compiler of Hybrid cc straightforwardly translates each statement A into an expression graph, where each node corresponds to an operator of the language. We optimize memory by sharing code as far as possible.

We omit a detailed description of the implementation, since most of it is based on standard techniques for how a concurrent constraint language based on reductions is implemented, e.g. we have borrowed from AKL and cc(FD) [HSD92, Jan94] in how constraints and suspensions are treated, how memory is managed (using a conservative garbage collector for C), and how backtracking is implemented (using choicepoint and trail stacks).

3 The constraint solvers

3.1 Nonlinear equations

We consider in the following only constraints of the form $f(\mathbf{x}) = 0$, as all other constraints can be reduced to this form by introducing slack variables. Interval pruning is used as the basic means for constraint solving. We have implemented four pruning operators: indexicals, interval splitting, the Newton-Raphson method, and the Simplex method. The pruning is hence stated as: given $f(x) = 0$ and an interval constraint $x \in [a, b]$ for x , apply one or more operators to $f(x)$ to compute a new interval $[a_1, b_1] \subseteq [a, b]$ for x . If this fails, the constraint is deemed inconsistent.

An *indexical* is the fastest way to update the interval for x [HSD92, Car95]. Given $f(x, \mathbf{y}) = 0$, we try rewriting the constraint in an explicit form $x = g(\mathbf{y})$, for some term g . Now x is set to $[a, b] \cap g(\mathbf{I}/\mathbf{y})$, where $\mathbf{y} \in \mathbf{I}$ holds in the current store, and g is evaluated over intervals.

For example, consider $x + y = 0$, $x \in [0, 3]$, and $y \in [-1, -2]$. Then the indexical $x = -y$ is used to set x to $[1, 2]$.

Splitting of intervals is used to narrow the interval for x by splitting it recursively. Given $f(x, \mathbf{y}) = 0$, $x \in [a, b]$, $\mathbf{y} \in \mathbf{I}$, we split $[a, b]$ until the smallest $a_1 \in [a, b]$ is found such that $0 \in f(a_1, \mathbf{I})$. Hence, if $0 \in f([a, \frac{a+b}{2}])$, then $a_1 \in [a, \frac{a+b}{2}]$, and otherwise $a_1 \in [\frac{a+b}{2}, b]$. Similarly, b_1 is computed, thus $x \in [a_1, b_1]$.

For example, given $x^2 = 1$ and $x \in [-\infty, \infty]$, it follows that $0 \in [-\infty, 0]$, but $0 \notin [-\infty, -100]$ say, so a_1 is determined to be in $[-100, 0]$. Eventually, a_1 is determined to be -1 .

The third pruning method is the *Newton-Raphson* method adapted to intervals [AH83, VMK95]. As in splitting, the leftmost and rightmost zeros for $f(x)$ are computed separately. Let $f'(x) = \frac{df(x)}{dx}$, $I = [a, b]$ such that $0 \in f(I)$ and $0 \notin f'(I)$ (this guarantees that there is only one zero in I , and can be accomplished by splitting), and let $m_i \in I_i$. Let $I_0 = I$, and define $I_{i+1} = I_i \cap (m_i - \frac{f(m_i)}{f'(I_i)})$. Iterate until $I_i = I_{i+1}$. It follows that $0 \in f(I_i)$.

In practice, we combine splitting and the Newton-Raphson method for quick results, just as in `clp(Newton)`. Splitting is useful in reducing the size of an interval, but is inefficient in pinning down the roots exactly. The Newton-Raphson method finds roots very quickly, if they are known to lie in a small interval. For example, given $x^2 = 1$ and $x \in I = [-\infty, \infty]$, we split I recursively until I is split down to $[-2, -1]$. By setting I_0 to I , and applying the Newton-Raphson method, $I_i = [-1, -1]$ is produced. Similarly, $[1, 1]$ is produced for the rightmost zero. Hence, the final interval returned by the pruning operator is $[-1, 1]$. Note that this interval is clearly an approximation to the set of solutions, since only -1 and 1 are solutions to the equation.

The *Simplex* method is used as a global pruning method and is applied only at certain times due to its high cost, unlike the previous lightweight methods, which are applied incrementally. It is useful for detecting inconsistent conjunctions that otherwise lead to slow convergence of the propagator, a well-known problem

when inequalities are used. For example, consider the conjunction $\{x \leq y - \epsilon, y \leq x - \epsilon\}$, for some small $\epsilon > 0$. This conjunction is inconsistent but the pruning algorithm reduces the size of the intervals by ϵ at each iteration, forcing a large number of iterations before an inconsistency is detected.

Given a set of constraints $C = \{c_1, \dots, c_k\}$, we linearize them into a set of linear constraints $L = \{l_1 = 0, \dots, l_k = 0\}$, by replacing each *nonlinear* term, e.g. xy , by a new variable z uniformly throughout the set C . The detection of common subexpressions is useful here. Hence, each l_i is of the form $a_0 + a_1 z_1 + \dots + a_n z_n = 0$, for some constants a_j and variables z_j . Note that C is consistent implies L is consistent.

Now, we apply the Simplex method to L to check whether L is inconsistent, using standard techniques. If successful, the Simplex algorithm can be used again for pruning the original variables of C . For each such variable x , a_1 (the new minimum value of x) is computed by minimizing x , and b_1 (the new maximum) by maximizing x .

By applying the above, a conjunction such as $x + y = 3, 2x - y = 0$ produces the interval constraints $x \in [1, 1]$ and $y \in [2, 2]$ immediately, whereas the other operators produce no pruning. The conjunction $x \leq y - \epsilon, y \leq x - \epsilon$ is shown to be inconsistent.

Implementation. Internally, each constraint c is decomposed into a set of pairs, (x, f) , where x is a variable in c and f is either the indexical that prunes x , derived as above, or c itself. The latter is then used for splitting and Newton-Raphson. Each variable y points to a set of such pairs, such that when y is pruned, the variables dependent on y are also pruned. Prunings are propagated by a variant of the arc-consistency algorithm (Figure 1).

We use an optimization based on the fact that decomposing a constraint as above produces equivalent variants, and hence when one variant is true the others are true too [Car95]. Hence, for each (x, f) that is dequeued, if f is marked as entailed, the pair is ignored since no more pruning is generated by f . When a pair (x, f) is enqueued, the list of variables of f is checked. If all of them (except the slack variables) are bounded by an interval $[a, a]$, f is marked entailed. All pairs generated from the same constraint share the entailment mark, hence when one is marked, they are all marked.

Telling in a point or interval phase. In a point phase, if we constrain the variable x' , then in addition to the propagation described above we infer that x is continuous, and set its value to the limiting value in the previous interval phase, if one exists.

In an interval phase, for an arithmetic constraint $t = 0$, for which t' is defined, the arithmetic constraint $t' = 0$ is also added (this is done recursively while the derivatives of all the variables are defined). For example, if $x + y = 0$ is added, then $x' + y' = 0$ is added too, if x' and y' are defined in the current state. This is sound, and improves the propagation and entailment checking by adding redundant constraints.

```

int propagate(queue) {
  while (queue is not empty) {
    (var,constraint) = dequeue(queue);
    if constraint is marked entailed continue;
    if constraint is an indexical
      interval = intersect(var,evalIndexical(constraint));
    else
      interval = project(var,constraint); // using splitting, NR
    if interval is empty return 0;
    if interval is a strict subset of var {
      var = interval;
      enqueue(var->constraints,queue);
      if all variables in constraint are determined
        mark constraint entailed;
    }
  }
  return 1;
}

```

Fig. 1. Pseudo-code for the propagator

3.2 Entailment checking

Let the current store be σ . In the point phase, a constraint $t = 0$ is entailed if t evaluates to $[0, 0]$ in σ , where each operator is evaluated over intervals rather than points, and where a variable x is replaced by $[a, b]$, where $x \in [a, b]$ belongs to σ . Similar reasoning is applied to $t \leq 0$ and $t < 0$. We assume that each arithmetic constraint is normalized to one of the forms above.

In the interval phase, if $t = 0$ was true in the previous point store, and $t' < 0$ holds in σ , then $t < 0$ also holds in σ . Similarly, for any positive natural number n for which $t^{(n)}$ is defined, if $t^{(m)} = 0$ and $t^{(n)} < 0$ are true in σ , for all m s.t. $0 < m < n$, $t < 0$ is true in σ . The constraint $t = 0$ is consequently true in the interval phase if $t^{(m)} = 0$ is ruled true for all m for which $t^{(m)}$ is defined.

3.3 Ordinary differential equations

We use a version of Runge-Kutta integration with adaptive step-size for integrating the differential equations numerically [PTVF92] (although it is easy to add other integration methods). The initial conditions of the integration are given by the store from the most recent point phase.

The pseudo-code for the integrator is shown in Figure 2, where we give the simpler fourth-order Runge-Kutta with no error checking to illustrate the interplay between propagation and integration.

We have made three changes to the basic Runge-Kutta algorithm. First, we use interval arithmetic throughout. This makes the system more flexible since

```

integrate(diff_eqs,check_list) {
  let h be the initial step size (0.1);
  let V be the dependent variables of diff_eqs;
  set the initial values of V by the store from the point phase;
  propagate;
next:
  integrate V;
  if a constraint in check_list is overshoot, backtrack and shrink h;
  if a constraint in check_list changes state, stop;
  go to next;
}

integrate V { // 4th order Runge-Kutta with no error-control
  compute k1 from current x' (for each x in V);
  for i in [2,4] {
    initialize a new store;
    update x (for each x in V) to compute ki;
    propagate;
    compute ki for current x' (for each x in V);
  }
  set new x = old x + 1/6*k1 + 1/3*k2 + 1/3*k3 + 1/6*k4;
  propagate; // to get values of variables after time-step
}

```

Fig. 2. Overview of the integration procedure

we do not require specific initial conditions, e.g. a variable x can be constrained to $[0, 100]$ at the start of the integration. However, the interval arithmetic also introduces a divergence problem. For some examples, the solution interval for a variable x grows in size as the integration proceeds, i.e. we lose precision. We are exploring methods for improving this automatically, but meanwhile we rely on using redundant constraints to contain the divergence (see example below).

Second, each step in the integration includes propagation (but with no use of the Simplex algorithm), so that we can solve simultaneous equations of an arbitrary form. In the standard Runge-Kutta procedure, x_{n+1} is computed from x_n (its previous value) by considering the explicit equation $x' = f(x)$ and computing x_{n+1} by: $x_{n+1} = x_n + \sum_{i=1}^6 c_i k_i$, and k_i is defined as: $k_1 = hf(x_n)$, $k_i = hf(x_n + b_{i1}k_1 + \dots + b_{i(i-1)}k_{i-1})$, $1 < i \leq 6$, where b_{ij} and c_i are constants given by the Runge-Kutta formulas, and h is the time step.

In Hybrid cc we do not necessarily have the equation $x' = f(x)$, but rather one or several equations on x' , e.g. $(x')^2 = x$. We thus compute k_i by first setting each dependent variable (x for which x' is constrained) to $x_n + b_{i1}k_1 + \dots + b_{i(i-1)}k_{i-1}$, where x_n is the previous interval for x , and then propagating, in an initially empty store σ_i , the consequences of the differential equations. At quiescence, k_i is set to $[ha_i, hb_i]$, where x' is constrained to $[a_i, b_i]$ in σ_i . Afterwards, each σ_i is discarded.

Example 1. The following system of equations describes the tank temperature(t) and concentration of a substance a (c_a) in a tank being stirred (the other parameters are constants) [Kay96].

$$\begin{aligned} c'_a &= \frac{c_{a_i} - c_a}{\tau} - k_0 e^{-E/t} c_a \\ t' &= \frac{t_i - t}{\tau} - k_0 e^{-E/t} c_a \end{aligned}$$

This system is highly nonlinear due to the exponential containing t . It diverges very quickly given an initial state such as $0.933 \leq c_a \leq 0.934$, $353.358 \leq t \leq 353.36$ — after some steps t and c_a both become $[0, \infty)$. Adding the constraints $c_a \geq 0.8445$ and $t' \leq 0$ to the set of differential equations keeps the intervals for c_a and t considerably narrower (the width being of the order 10^{-2}). These constraints can be obtained automatically by using qualitative methods [Kay96].

Third, the integration is made to interrupt *exactly* at the time instant when some constraint in a given list of constraints to be checked changes its state. This is necessary to make the results of the implementation be independent of the step-size, modulo numerical errors. Thus in the program 1, $x = 2$ is false while $x \in (0, 2)$, but when the integrator reaches $x = 2$ we must switch to the point phase. The point when the integrator stops is called the *breakpoint*.

The standard Runge-Kutta procedure however does not know about the breakpoints, so it may overshoot, e.g. in the program above, go from $x = 1.9$ to $x = 2.3$ in one step, depending upon the current integration stepsize. Hence, we must force the integrator to consider every “important” point.

We detect overshooting by recording, for each given constraint to be checked, whether it is initially true or false or neither. For each integration step, we check whether the status of any constraint changes, e.g. goes from true to false. When we detect overshooting, we backtrack, i.e. we undo the most recent integration step, and try a smaller step size to find the point when the break should happen exactly, e.g. in the program above, force the integrator to reach $x = 2$. Currently, we use a simple linear interpolation technique for computing the smaller step size, though more sophisticated techniques are possible.

4 Examples and Evaluation

We now give an idea of the performance of Hybrid cc on some representative benchmarks picked from [vHLB97]. We show the runtimes of Hybrid cc computing all the solutions to each problem on a SPARCstation 20 system. For example, the Broyden Banded functions are computed by the following constraints:

$$f_i(x_1, \dots, x_n) = x_i(2 + 5x_i^2) + 1 - \sum_{j \in J_i} x_j(1 + x_j) \quad (1 \leq i \leq n)$$

where $J_i = \{j \mid j \neq i \& \max(1, i - 5) \leq j \leq \min(n, i + 1)\}$, which for $n = 3$ is written in Hybrid cc as:


```

0 = x1 * (2 + 5*x1^2) + 1 - x2*(x2+1),
0 = x2 * (2 + 5*x2^2) + 1 - x1*(x1+1) - x3*(x3+1),
0 = x3 * (2 + 5*x3^2) + 1 - x1*(x1+1) - x2*(x2+1),
-1 <= x1, x1 <= 1, -1 <= x2, x2 <= 1, -1 <= x3, x3 <= 1

```

For these problems, the constraint solver of Hybrid cc finds the unique solution to within 10^{-6} .

The other examples we have considered are the Moré-Cosnard nonlinear integral equations, an interval arithmetic problem (i4), and a combustion problem [vHLB97]. We give the runtimes for some different n in the case of the Broyden Banded, and the Moré-Cosnard equations.

Example	run-time (sec)	Example	run-time (sec)
Broyden 10	0.13	Moré-Cosnard 40	39.8
Broyden 40	0.6	Moré-Cosnard 80	436
Broyden 160	2.6	interval 4	21.2
Moré-Cosnard 10	0.4	combustion	6.2

These numbers compare with the numbers published for clp(Newton) as follows. In the Broyden and Moré-Cosnard, Hybrid cc is between 3 and 5 times as fast, taking the difference between the hardware used into account. For the interval-4 example, Hybrid cc is twice as fast as clp(Newton), and for the combustion example 50% slower.

We present a longer example illustrating the use of Hybrid cc in modeling a hybrid system. The scenario modeled is a pool table with several balls rolling on it with various initial velocities. The balls keep rolling in a straight line until they hit another ball or the edge of the table or fall into a pocket, or come to rest due to friction.

The class `Ball` defines a ball with initial parameters giving its position and velocity. Its properties are its position and velocity, and some signals to notify changes in its velocities etc. The initial velocity and position is set up. The direction of motion of the ball is also computed as $\cos^2 \theta$, where θ is the direction of motion of the ball. The ball is active until it falls into a pocket, indicated by `trap Pocketed` in ... — at that moment all program fragments associated with the ball are terminated. While the ball is rolling, its velocity decreases according to friction. If `ChangeX` or `ChangeY` become true, then the program issuing the Change signal computes the new velocity. `lcont(x)` asserts that the variable `x` is left continuous, and `rcont(x)` asserts that the variable `x` is right continuous.

The closures `Edge` and `Collisions` keep checking if the balls collide with the edge of the table or with each other. In each case a new velocity is computed for the ball(s) involved, according to the standard laws of kinematics. The closure `Pocketed` checks if any ball has fallen into a pocket, and issues the appropriate signal to terminate the ball's existence.

```

Ball = (initpx, initpy, initvx, initvy)
      [px, py, vx, vy, ChangeX, ChangeY, Pocketed] {
      px = initpx, py = initpy,

```

```

vx = initvx, vy = initvy,
new direction in {
  direction = vx^2/(vx^2 + vy^2),
  do always {
    cont(px), cont(py),
    lcont(vx), lcont(vy),
    if (ChangeX || ChangeY) then direction = vx^2/(vx^2 + vy^2),
    unless (ChangeX || ChangeY) then direction' = 0,
    px' = vx, py' = vy,
    unless ChangeX then {
cont(vx),
if (vx < 0) then vx' = fric * direction^0.5,
if (vx > 0) then vx' = -fric * direction^0.5,
if (vx = 0) then vx' = 0
    },
    unless ChangeY then {
cont(vy),
if (vy < 0) then vy' = fric * (1 - direction)^0.5,
if (vy > 0) then vy' = -fric * (1 - direction)^0.5,
if (vy = 0) then vy' = 0
    }
  } watching Pocketed
},
Edges = (){
  always forall Ball(X) do {
    if (X.px = radius || X.px = xMax - radius) then {
      XEdgeCollision,
      X.ChangeX, X.vx = -prev(X.vx)
    },
    if (X.py = radius || X.py = yMax - radius) then {
      YEdgeCollision,
      X.ChangeY, X.vy = -prev(X.vy)
    }
  }
},
Collisions = (){
  always forall Ball(A) do forall Ball(B) do
    if (A < B) then //not the same ball
      if ((B.px - A.px)^2 + (B.py - A.py)^2 = 4*radius^2) then {
Collision,
if (A.px = B.px) then {
  A.ChangeY, B.ChangeY,
  A.vy = prev(B.vy),
  B.vy = prev(A.vy)
}
}
}

```

```

},
if (A.px < B.px) then {
  A.ChangeX, B.ChangeX,
  A.ChangeY, B.ChangeY,
  new c in new ix in {
    c := (A.py - B.py)/(A.px - B.px),
    ix := (prev(B.vx - A.vx) + c*prev(B.vy - A.vy))/(1+c^2),
    B.vx = prev(B.vx) - ix,
    B.vy = prev(B.vy) - c*ix,
    A.vx + B.vx = prev(A.vx + B.vx), // X-momentum
    A.vy + B.vy = prev(A.vy + B.vy) // Y-momentum
  }
}
}
},
Pockets = (){
  always forall Ball(X) do
    if (prev(X.px)^2 + prev(X.py)^2 <= pocket^2
      || prev(X.px)^2 + (prev(X.py)-yMax/2)^2 <= pocket^2
      || prev(X.px)^2 + (prev(X.py)-yMax)^2 <= pocket^2
      || (prev(X.px)-xMax)^2 + prev(X.py)^2 <= pocket^2
      || (prev(X.px)-xMax)^2 + (prev(X.py)-yMax/2)^2 <= pocket^2
      || (prev(X.px)-xMax)^2 + (prev(X.py)-yMax)^2 <= pocket^2)
    then
      X.Pocketed
},
always { radius = 3, xMax = 150, yMax = 300, pocket = 7, fric = 1},
Ball(B1, 10, 10, 25, 25),
Ball(B2, 20, 11, -35, 55),
Ball(B3, 80, 51, -15, 49),
Edges(),
Collisions(),
Pockets()

```

The last few lines set up the initial configuration. We ran this program for 74 simulated time units, after which all the balls were either at rest or pocketed — the total time of execution was 0.77 seconds on an UltraSparc 2.

5 Related work

The SHIFT programming language developed at UC Berkeley [DGS96, SDG96] is also intended for simulation of hybrid systems. Programs in SHIFT are synchronous concurrent collections of hybrid automata [ACH⁺95]. Computations proceed in alternating point and interval phases, as in Hybrid cc. SHIFT has an object-oriented framework for constructing models, and also has constructs

with side-effects which are useful in writing state-machines. However it is not a declarative language — the transitions and states have to be explicitly programmed. The interaction of concurrency and side-effects also causes semantic problems in maintaining determinism. In the current implementation of SHIFT, only fixed step-size Runge-Kutta integration is supported, and breakpoints can occur only at the end of a step.

Differential equations with intervals are an active field of research, we will not attempt to provide a survey here. Most of the research is concerned with using intervals to provide validated solutions to differential equations, *i.e.* a statement of the form that the solution must lie in a certain interval. For a starting point into the field, see the tutorial by George Corliss [Cor95].

The field of interval reasoning is even larger. Several systems for interval constraint solving have been built, one of the recent ones is `clp(Newton)` [VMK95, vHLB97], which uses similar propagation methods, but also exploits multiple representations of constraints. This naturally leads to better pruning of the intervals in many problems. However the above comparison shows that in many problem instances, the performance of our system is comparable to that of `clp(Newton)`.

6 Conclusion and Future Work

We have presented an implementation of a programming language for hybrid systems, `Hybrid cc`. The key feature of the language, which is reflected in the implementation, is that it is constraint-based, using interval constraints. The interval constraints are necessary to make `Hybrid cc` applicable to many modeling problems, and the interval propagation used inside the numerical solver for differential equations improves the accuracy of the integration.

`Hybrid cc` will be used to construct real-life models for engineering and educational purposes. We have already started some work in this direction, and plan to use `Hybrid cc` for simulation of rovers and spacecraft.

References

- [ACH⁺95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.
- [AH83] Gotz Alefeld and Jurgen Herzberger. *Introduction to Interval Computations*. Academic Press, 1983.
- [BB91] A. Benveniste and G. Berry. The synchronous approach to reactive and real-time systems. *Proceedings of the IEEE*, 79(9):1270–1282, September 1991.
- [Car95] Bjorn Carlson. *Compiling and Executing Finite Domain Constraints*. PhD thesis, Uppsala University, 1995.
- [Cor95] G. F. Corliss. Guaranteed error bounds for ordinary differential equations. In W.A.Light and M.Marletta, editors, *Theory of Numerics in Ordinary and Partial Differential Equations*, volume IV of *Advances in Numerical Analysis*, pages 1–75. Oxford University Press, 1995.

- [DGS96] Akash Deshpande, Aleks Gollu, and Luigi Semenzato. The SHIFT programming language and run-time system for dynamic networks of hybrid automata. Technical report, UC Berkeley PATH Project, 1996. www-path.eecs.berkeley.edu/shift/doc/ieeshift.ps.
- [GJS97] Vineet Gupta, Radha Jagadeesan, and Vijay Saraswat. Computing with continuous change. *Science of Computer Programming*, 1997. To appear. Available from <http://ic.arc.nasa.gov/people/vgupta>.
- [GJSB95] Vineet Gupta, Radha Jagadeesan, Vijay Saraswat, and Daniel Bobrow. Programming in hybrid constraint languages. In Panos Antsaklis, Wolf Kohn, Anil Nerode, and Sankar Sastry, editors, *Hybrid Systems II*, volume 999 of *Lecture notes in computer science*. Springer Verlag, November 1995.
- [GSS95] Vineet Gupta, Vijay Saraswat, and Peter Struss. A model of a photocopier paper path. In *Proceedings of the 2nd IJCAI Workshop on Engineering Problems for Qualitative Reasoning*, August 1995.
- [Hal93] N. Halbwachs. *Synchronous programming of reactive systems*. The Kluwer international series in Engineering and Computer Science. Kluwer Academic publishers, 1993.
- [Har87] D. Harel. Statecharts: A visual approach to complex systems. *Science of Computer Programming*, 8:231 – 274, 1987.
- [HSD92] Pascal Van Hentenryck, Vijay A. Saraswat, and Yves Deville. Constraint processing in cc(fd). Technical report, Computer Science Department, Brown University, 1992.
- [Jan94] S. Janson. *AKL – A Multiparadigm Programming Language*. PhD thesis, Uppsala University, 1994.
- [Kay96] Herbert Kay. *Refining Imprecise Models and Their Behaviors*. PhD thesis, University of Texas at Austin, 1996.
- [PTVF92] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 1992.
- [Rei80] Ray Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81 – 132, 1980.
- [Sar93] Vijay A. Saraswat. *Concurrent constraint programming*. Doctoral Dissertation Award and Logic Programming Series. MIT Press, 1993.
- [SDG96] Luigi Semenzato, Akash Deshpande, and Aleks Gollu. The SHIFT reference manual. Technical report, UC Berkeley PATH Project, 1996. www-path.eecs.berkeley.edu/shift/doc/shift.ps.gz.
- [SJG96] V. A. Saraswat, R. Jagadeesan, and V. Gupta. Timed Default Concurrent Constraint Programming. *Journal of Symbolic Computation*, 22(5-6):475–520, November/December 1996. Extended abstract appeared in the *Proceedings of the 22nd ACM Symposium on Principles of Programming Languages*, San Francisco, January 1995.
- [vHLB97] Pascal van Hentenryck, Michel Laurent, and Frederic Benhamou. Newton: Constraint programming over non-linear constraints. *Science of Programming*, 1997. to appear.
- [VMK95] Pascal Van Hentenryck, David McAllester, and D. Kapur. Solving polynomial systems using a branch and prune approach. *SIAM Journal of Numerical Analysis*, 1995. (Accepted). (Also available as Brown University technical report CS-95-01.).

Reachability Analysis via Face Lifting*

Thao Dang and Oded Maler

VERIMAG, Centre Equation, 2, av. de Vignate, 38610 Gières, France,
{Thao.Dang, Oded.Maler}@imag.fr

Abstract. In this paper we discuss the problem of calculating the reachable states of a dynamical system defined by ordinary differential equations or inclusions. We present a prototype system for approximating this set and demonstrate some experimental results.

1 Introduction

One of the main activities in verifying a discrete system consists in finding the set of system states which are reachable, via the transition relation, from a given initial set of states (control synthesis for discrete-event systems [RW89] can ultimately be reduced to some variant of reachability analysis [AMP95-b]). For small finite-state systems this is done using simple graph algorithms which manipulate set-theoretical representations of the reachable sets. For systems which are very large, or even infinite, symbolic methods are used, that is, the set of states reachable after k steps of the system is represented by some formula rather than being enumerated explicitly.

Some of this technology has been exported to certain classes of hybrid systems which deserve to be termed *piecewise-trivial dynamical systems*. These systems, such as timed automata [AD94] or PCD systems¹ [ACH⁺95], [AMP95-a] exhibit a trivial dynamics in the continuous phase, and all their complexity is due to the interaction between this dynamics and the discrete transitions. For such systems, given some initial polyhedral subset of the state-space, the sets of all its successors via the continuous dynamics can be calculated by straightforward linear algebraic calculation. Even with this simplicity, the reachability problem for such systems is undecidable or even worse ([HKPV95], [AM95]). A practical conclusion from the experience with this class of systems is not to look for fully-automatic decision procedures but rather for more modest goals while trying to analyze continuous systems.

In this paper we discuss the problem of extending the methodology of calculating reachable sets to systems with non-trivial continuous dynamics and no discrete dynamics at all,² namely systems defined by ordinary differential

* This research was supported in part by the European Community project HYBRID EC-US-043. VERIMAG is a joint laboratory of CNRS and UJF.

¹ Dynamical systems with piecewise-constant derivatives; The term *Linear Hybrid Automata* used in [ACH⁺95] is unfortunate and causes confusion with linear systems.

² Discrete transitions can later be incorporated naturally into the continuous techniques, if and when such techniques are established.

equations. We formulate the problem and describe a technique, suggested by M. Greenstreet [G96], for over-approximating reachable sets. We then introduce a variation on this technique which can be applied more easily to more than two dimensions. Finally we show the results obtained by an experimental implementation of the algorithm for both linear and non-linear systems.

2 Statement of the Problem

2.1 Deterministic Systems

Definition 1 [Dynamical System.] A differential dynamical system is $S = (X, f)$ where $X = \mathbb{R}^n$ is the Euclidean space and $f : X \rightarrow X$ is a continuous function (vector field). A behavior of S starting from a point $x_0 \in X$ is a trajectory $\xi : \mathbb{R}_+ \rightarrow X$ satisfying $\xi[0] = x_0$ and for every t ,

$$d\xi[t]/dt = f(\xi[t]).$$

People less pedantic than the average formal methodologists would simply say:

$$\dot{x} = f(x).$$

It can also be expressed in a somewhat more operational manner:

$$\xi[t] = x_0 + \int_0^t f(\xi[\tau])d\tau.$$

The set of states reachable by the system from x_0 is defined as

$$Reach(x_0, f) = \{\xi[t] : t \geq 0\}.$$

Typically when we want to prove safety properties of such a system we would like to show that $Reach(x_0, f) \cap Q = \emptyset$ for some $Q \subseteq X$. Except for the rare case when $Reach(x_0, f)$ has a closed-form solution, such as $\{x_0 e^{At} : t \in \mathbb{R}_+\}$ for linear systems, the common way to achieve that goal is to use numerical integration to calculate an approximation of $Reach(x_0, f)$ incrementally. This means starting from $\xi[0] = x_0$ and applying some iteration

$$\xi[(n+1)\Delta] = \xi[n\Delta] + g(\xi[n\Delta])$$

where Δ is the discretization step and g is supposed to be a good approximation of the integral.

According to the strict standards of discrete verification, this approach is far from being satisfactory: first, we compute ξ only for a small subset of time points, and we might miss a visit of the system in Q at some t , $n\Delta < t < (n+1)\Delta$. Secondly, even for points of the form $t = n\Delta$, we compute only an approximation of $\xi[t]$. And finally, the calculation is not guaranteed to terminate (and if it terminates, it is not always for a good reason). Termination of the calculation

of $Reach(x_0, f)$ means that the trajectory becomes periodic,³ i.e. $\xi[t] = \xi[t']$ for some $t' > t$, which may sometimes happen numerically only because we approximate the ideal mathematical reals by a finite subset of the rationals. Nevertheless, generations of mathematicians, pure and applied, assure us that given reasonable f and Q , we can find Δ and g such that we need not worry about the first two problems. As for the third one, we should accept it as a sad fact of life, as do all engineers who use simulation methods.

To summarize, given a system (X, f) , an initial state x_0 and a set of bad states Q , we have a methodology, or a semi-algorithm (modulo some numerical conditions) for verifying that from x_0 you never reach Q :

```

 $R_0 := \{x_0\};$ 
repeat       $i = 1, 2 \dots$ 
               $R_i := R_{i-1} \cup Next(R_{i-1})$ 
until       $(R_i = R_{i-1}) \vee (R_i \cap Q \neq \emptyset) \vee (\text{The user gives up})$ 

```

Here, $Next(R_i)$ means just integrating numerically starting from the last element of R_i . Up to this point this is nothing but rephrasing, in a somewhat awkward manner, the common practice of simulation.

2.2 Non-deterministic Systems

In many situations we cannot be sure of the initial conditions nor of the dynamics of the system. In most cases we will have an equation of the form

$$\dot{x} = f(x, u).$$

where u is some unobserved external disturbance, about which we know only some constraints.⁴ The behavior of the system resulting from interaction with any admissible input u can be characterized using *differential inclusion* [AC84] of the form

$$\dot{x} \in F(x),$$

where $F : X \rightarrow 2^X$ is roughly

$$\bigcup_u f(x, u).$$

This is the continuous analogue of a non-deterministic transition system. Such a system, when started at some initial state x_0 , usually produces dense bundles of trajectories (solutions), which we denote by $L(F, x_0)$. The set of states reachable from x_0 at time t (which was simply $\{\xi[t] : t \in \mathbb{R}_+\}$ in deterministic systems) is defined as

$$Reach_t(x_0, F) = \bigcup_{\xi \in L(F, x_0)} \xi[t].$$

³ Which is always the case in finite-state systems.

⁴ Things get even more complicated in control synthesis problems whose generic form is $\dot{x} = f(x, u, v)$ where u and v are two different types of external inputs.

The set of all states visited during the interval $[0, t]$ is

$$Reach_{[0,t]}(x_0, F) = \bigcup_{\tau \in [0,t]} Reach_{\tau}(x_0, F)$$

and the set of all reachable states is

$$Reach(x_0, F) = Reach_{[0,\infty]}(x_0, F).$$

In order to apply the symbolic verification methodology we would like to have a diverging sequence t_0, t_1, \dots of time points and calculate a sequence $R_0, R_1 \dots$ such that $R_0 = \{x_0\}$ and for every i , $R_i = Reach_{[0,t_i]}(x_0, F)$. As in the case of numerical integration of a single trajectory, the calculation of R_{i+1} will be based on f and R_i , and from a computational viewpoint, the main novel feature here is the calculation of differential successors of *a set of points* rather than that of a *single point*. This motivates us to attack first a slightly more restricted version of the problem: calculating the reachable states of a *deterministic* system starting from a *set* $P \subseteq X$, namely to find

$$Reach(P, f) = \bigcup_{x \in P} Reach(x, f).$$

This problem already exhibits the major computational difficulty associated with representing and simulating a set of trajectories (see figure 1 for an illustration of the above notions).

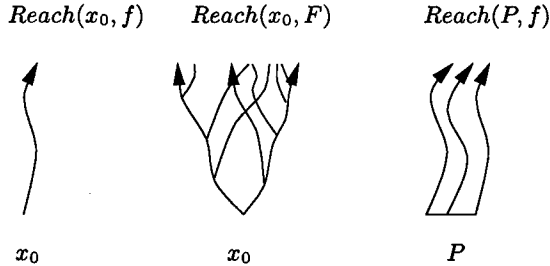


Fig. 1. Calculating reachable states for: 1) A deterministic system starting at a point, 2) A non-deterministic system starting at a point and 3) A deterministic system starting at a set.

3 The Face Lifting Approach

We assume from now on that everything takes place inside a bounded subset of X in which f is Lipschitz.

3.1 Arbitrary Polyhedra

The first ingredient of any solution is a formalism for representing subsets of X . Not being computer algebraists, we restrict ourselves to polyhedral sets. These are sets which can be written as boolean combinations of linear inequalities.⁵ Polyhedral sets come in two major varieties, convex and non-convex. Those of the former type can be written as conjunctions of inequalities (intersections of half-spaces) and they are uniquely determined by their sets of vertices.

If the initial set P is convex and f preserves convexity (as in the case of linear systems), we are lucky because for every t we have

$$Reach_t(conv(x_1, \dots, x_n), f) = conv(Reach_t(x_1, f), \dots, Reach_t(x_n, f))$$

where $conv$ denotes the convex hull. With this property it would have been sufficient to simulate a finite number of trajectories starting at the vertices. However, in the case of arbitrary differential systems, the approximation of a non-convex polyhedron by its convex hull is usually useless. Just consider what such an approximation gives when P contains a bifurcation point.

The treatment of non-convex polyhedra poses enormous problems in terms of representation, normal forms (which are important to detect the condition $R_{i+1} = R_i$), etc. In the sequel we present a technique, due to M. Greenstreet [G96], which we call *face lifting*. In the abstract sense, face lifting can be applied to systems in *any* dimension, but concretely, its practical application to 3 or more dimensions is not at all evident.

The approach is based, first of all, on the following basic observation concerning continuous trajectories: if some point $y \in Reach_t(x, f) - P$ for an interior point $x \in P$, then there exists a point $x' \in bd(P)$ (the boundary of P) and $t' < t$ such that $y \in Reach_{t'}(x', f)$. In other words,

$$Reach_{[0,t]}(P) = P \cup Reach_{[0,t]}(bd(P)).$$

Hence, when coming to calculate R_{i+1} from R_i it is sufficient to look at the boundary of the latter (the union of its *faces* in the case of polyhedral sets and, in particular, its *edges* in 2-dim).

Consider a face e of a polyhedron such that it is included in the set characterized by the linear equality $a \cdot x = b$. Let $\hat{f}_e(x)$ denote the *outward component* of $f(x)$ relative to e , that is, the projection of $f(x)$ on the normal to e , and let $\hat{f}(e)$ denote its maximum over $x \in N(e)$, where $N(e)$ is some neighborhood of e . Clearly, if $\hat{f}(e)$ is negative, the face does not contribute new reachable states which cannot be reached from other faces. Otherwise, for every Δ , one can find an ε such that all the points reachable from e in time Δ satisfy

$$a \cdot x \leq b + \Delta \cdot \hat{f}(e) + \varepsilon.$$

Geometrically speaking, this amounts to lifting the face e outward by $\Delta \cdot \hat{f}(e) + \varepsilon$ (see figure 2). (We omit some details concerning the relation between $\Delta, N(e)$,

⁵ If you want to impress non-logicians, you can say they are possible models of sentences in the first order theory of $(\mathbb{R}, +, <)$ or something.

ε and the Lipschitz constant of f , which guarantees the desired property of the approximation). This gives the following procedure for over-approximating $\text{Reach}_{[0,\Delta]}(P, f)$:

Calculate $\hat{f}(e)$ for every face e of P . Based on these find the appropriate ε and push every e whose $\hat{f}(e)$ is positive by $\Delta \cdot \hat{f}(e) + \varepsilon$ to obtain P' .

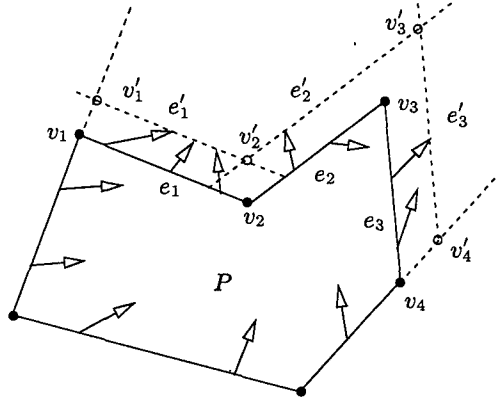


Fig. 2. A 2-dimensional example of the approach: a polyhedron P and a sample of the values of f on its edges. Only edges e_1 , e_2 and e_3 have a positive outward component of f and they are pushed into e'_1 , e'_2 and e'_3 . The vertices $\{v_1, \dots, v_4\}$ are replaced by $\{v'_1, \dots, v'_4\}$.

By construction, we have $\text{Reach}_{[0,\Delta]}(P, f) \subseteq P'$. It can be shown that locally, you can make the difference between the reachable set and its approximation as small as you like, by taking smaller Δ . Better approximation can be achieved by cutting a face into sub-faces whenever \hat{f} has a large variation over the face. However, there are cases where, in the long run, the method will produce unboundedly large over-approximations of $\text{Reach}(P, f)$, as shown in figure 3.

We have implemented the method for dimension 2 and obtained results similar to those obtained by other means (see section 4 for experimental results). However the extension to more than two dimensions is difficult as the special properties of the plane no more hold. In \mathbb{R}^2 , an ordered set of vertices always defines a unique polygon⁶ and the abstract operation of identifying a face can be realized by picking a pair of neighboring vertices. Similarly, the face lifting operation can ultimately be realized by replacing vertices in a list.

This is not true in more than two dimensions, where even convex polyhedra can exhibit a complicated structure with degeneracy which makes face recognition very hard. Consequently, we have tried another approach, slightly inspired by the basic ideas underlying the numerical solution of PDEs.

⁶ In fact, if we do not insist on *connected* polygons, it defines either the polygon or its complement.

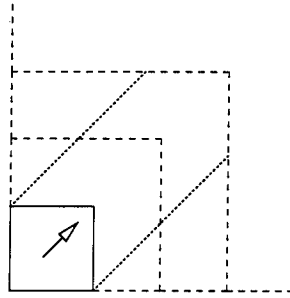


Fig. 3. A bad example: consider an axes-parallel rectangle and a constant vector field f with non-zero components in both dimensions. The reachable set lies between the two dotted diagonal lines, but the method will produce the whole upper-left orthant.

3.2 Griddy and Isothetic Polyhedra

Consider the sub-class of polyhedra which can be obtained by boolean combinations of inequalities of the form $x_i \leq c$ where x_i is a component of x and c is an integer constant.⁷ In other words, we partition the space into uniform hyper-rectangles and consider all polyhedra which can be written as unions of those (see figure 4-a). We call these *griddy polyhedra*.

Since such polyhedra are “finitely generated” (in a bounded sub-space) they admit a very simple representation using n -dimensional 0–1 matrices. It is also easy to determine whether an $(n-1)$ -dimensional hypercube is indeed part of the face of the polyhedron, and there is a systematic simple way to enumerate all the faces and calculate \hat{f} , which is now always parallel to one of the axes (see figure 4-a). With such a representation we can apply, in principle, face lifting in *any* dimension.

Techniques developed for griddy polyhedra can be adapted to the more general class of *isothetic* polyhedra, generated by arbitrary axes-parallel hyper-rectangles. These can be represented by a non-uniform grid depending on the represented polyhedron. The set of grid coordinates in any dimension consists of all projections of vertices of the polyhedron (see figure 4-b) and may change during the computation. The non-uniform grid has two main advantages over the uniform one:

1. Space: a griddy polyhedron which can be decomposed into few large rectangles can be represented more succinctly. However, when this method is used to represent, say, an approximation of a circle, the grid becomes very dense and this advantage is lost.

⁷ Of course, c can belong to the set of integer multiples of some rational constant as well.

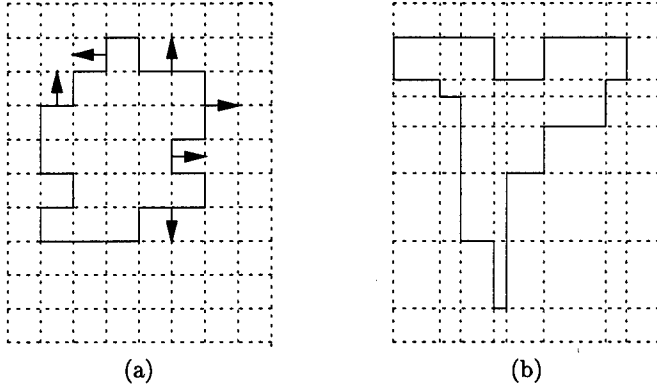


Fig. 4. (a) A Griddy Polygon. Some of the faces are annotated by their corresponding outward directions. (b) An isothetic polygon and its associated non-uniform grid. Face lifting can cause a refinement of the grid.

2. Expressive power and accuracy: with a fixed grid we need to push every face further to the next integer value, which sometimes creates an unnecessary over-approximation, beyond what is inherent in face lifting alone (see example in the next section). With a variable grid we can push faces as little as we want.

Both methods are not very space efficient and we are currently investigating a canonical and much more succinct representation of these polyhedra.

4 Experimental Results

We have implemented griddy face lifting in 2 and 3 dimensions using the above-mentioned representation methods. For the uniform grid we use simply an n -dimensional array. For the non-uniform grid we use a linked list representation which currently consumes much more computation time.

In both methods we decompose every face into elementary hyper-rectangular elements and apply the basic operation of numerical optimization of \hat{f} to every such element. This is, of course, less efficient than a coarser decomposition of the face into larger hyper-rectangles, an approach we intend to implement in the future. On the other hand, this is better in terms of accuracy. All the results described below, except for the 3-dimensional example, were obtained using the fixed grid implementation.

4.1 Linear Systems in \mathbb{R}^2

In figure 5 we demonstrate the behavior of the algorithm on various classes of linear systems of the form $\dot{x} = Ax$ (see [HS74] for the classification). We treat the following cases:

Type	A	Initial set
<i>Center</i>	$\begin{pmatrix} 0.0 & -6.0 \\ 3.0 & 0.0 \end{pmatrix}$	$[-0.25, 0.25] \times [-0.25, 0.25]$
<i>Node</i>	$\begin{pmatrix} -5.0 & 0.0 \\ 0.0 & -2.0 \end{pmatrix}$	$[0.2, 0.5] \times [0.2, 0.4]$
<i>Saddle</i>	$\begin{pmatrix} -5.0 & 0.0 \\ 0.0 & 4.0 \end{pmatrix}$	$[0.0, 0.4] \times [-0.0, 0.4]$
<i>Sink</i>	$\begin{pmatrix} -2.0 & -3.0 \\ 3.0 & -2.0 \end{pmatrix}$	$[-0.1, 0.3] \times [0.1, 0.3]$

Sometimes, the use of a fixed grid generates an over-approximation which covers all the space. This is evident in the case of a *center* where every edge will have a non-zero outward component in some dimension.⁸ Consequently we have changed in these cases the rounding rule to obtain the desired result, that is, we push a face to the nearest grid unit and not necessarily outward. The price is in not being an over-approximation anymore. Using a variable grid is another way to solve this problem. Note that optimization of a linear \hat{f} is much cheaper computationally in the linear case.

4.2 Mixing Tank

This example, taken from [SKE97], is a typical non-linear equation encountered in chemical engineering. The variables x_1 , x_2 denote, respectively, the height and the concentration of liquid in a mixing tank with two inlets (with different rates and concentrations) and one outlet. The equation is

$$\dot{x}_1 = a_1 - a_2\sqrt{x_1}$$

$$\dot{x}_2 = \frac{1}{a_3 x_1}(1 - a_4 x_2)$$

With our choice of parameters, (1.322, 1.652) is an equilibrium state of the system. In figure 6 the states reachable from an initial set $[1.12 \times 1.17] \times [1.56 \times 1.68]$ are depicted, and one can see the convergence to the equilibrium.

⁸ At least, this case is not generic.

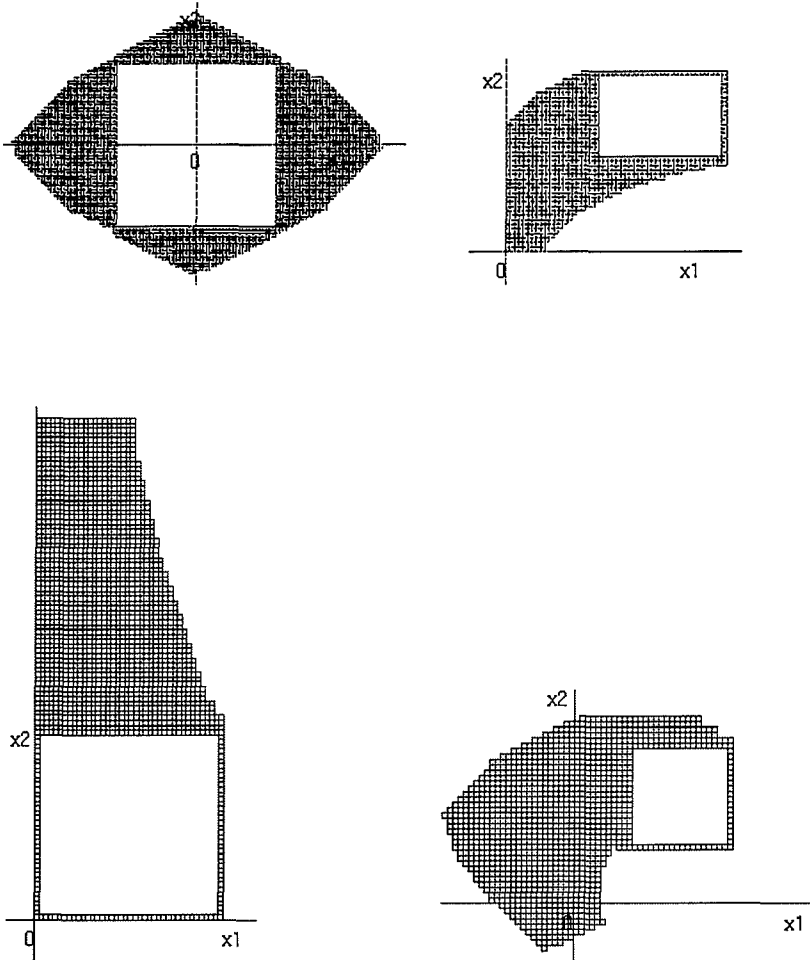


Fig. 5. Reachable sets of linear systems of type: 1) Center, 2) Node, 3) Saddle and 4) Sink. The white rectangles denote the initial sets.

4.3 Airplane Safety

The next example is taken from [LTS97]. The state variables x_1 , x_2 represent, respectively, the velocity and the flight path angle. Their evolution is governed by

$$\dot{x}_1 = -\frac{a_D x_1^2}{m} - g \sin x_2 + \frac{u_1}{m}$$

$$\dot{x}_2 = \frac{a_L x_1 (1 - c x_2)}{m} - \frac{g \cos x_2}{x_1} + \frac{A_L c x_1}{m} u_2$$

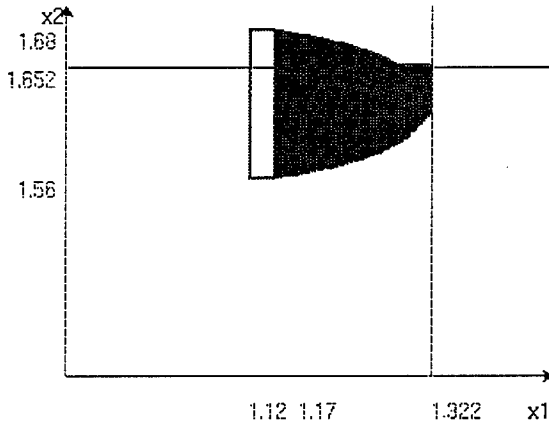


Fig. 6. Mixing Tank

The problem is to determine the safe subset of the state-space, i.e. the states from which the system does not leave the envelope P defined as the rectangle $[V_{min}, V_{max}] \times [\theta_{min}, \theta_{max}]$. This is equivalent to calculating the complement of the set of states reachable from $X - P$ by the reverse system. The results, depicted in figure 7 correspond to specific choices of values for parameters and for the controls $u_1 = \theta_{min}, u_2 = T_{max}$ (left) and $u_1 = \theta_{max}, u_2 = T_{min}$ (right). The results are consistent with those obtained in [LTS97].

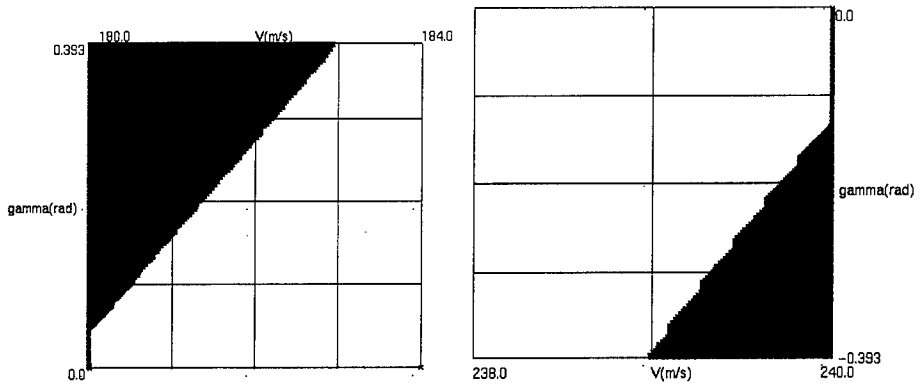


Fig. 7. Airplane Safety

4.4 Linear Systems in \mathbb{R}^3

In figure 8 one can see the reachable set of a 3-dimensional system with

$$A = \begin{pmatrix} -2 & 0 & 0 \\ 1 & -2 & 0 \\ 0 & 1 & -2 \end{pmatrix}$$

starting from the initial region $[-0.025, 0.025] \times [-0.1, 0.1] \times [0.05, 0.07]$.

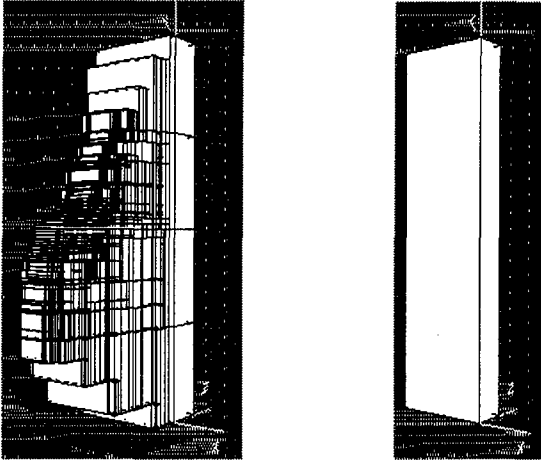


Fig. 8. Reachable states (left) starting from an initial region (right) for a 3-dimensional linear system.

5 Relation to other Work

There are various works concerning the calculation of reachable sets for differential inclusions. Many of these works are numerical analytic in nature, concerned mostly with calculation of abstract error bounds and less with the crucial questions of data-structures for high dimensional sets.

The problem of calculating $Reach(P, f)$ can be rephrased as a PDE⁹

$$\frac{\partial \varphi}{\partial t} = -\text{grad}(\varphi) \cdot f$$

where $\varphi : X \times \mathbb{R}_+ \rightarrow \{0, 1\}$ is defined as $\varphi(x, t) = 1$ iff $x \in Reach_{[0, t]}(P, f)$ and in particular $\varphi(x, 0) = 1$ iff $x \in P$. Sometime a “continualized” version of φ is

⁹ We owe this insight to P. Caspi [C93]. See also [TPS98] for a PDE-based approach.

used, namely a function $\varphi : X \times \mathbb{R}_+ \rightarrow \mathbb{R}$ such that $\varphi(x, 0) = 0$ exactly when x is on the boundary of P and $f(x, 0) > 0$ if x is inside P . Various methods exist for tracking the evolution of φ , see, e.g. [S96]. So far we have found no special computational nor didactic advantage in viewing the problem as a PDE instead of a direct ODE formulation, but this might change in the future.

In [PBV96] an alternative approach was suggested based on cutting the state-space into cubes, and associating with every cube a *rectangular differential inclusion* which is a differential inclusion of the form $c_i < \dot{x}_i < d_i$ for every i , with constants c_i and d_i . The reachability problem is decidable for this class of systems [PV94], and the idea here is to do *exact* calculations on an *approximate* model, where the bounds on f are calculated in a preprocessing stage. Similar to face lifting, this approach can guarantee, by refining the grid, error bounds only for a *finite* time horizon. This approach has been applied to several examples in [HW96] and in [SKE97]. Some of the ideas underlying face lifting appear already in [KM91] where the authors try to prove a homomorphism from a transistor-level differential model into an automaton. While doing so they also cut the space into a grid and try to calculate the reachability relation among cubes.

Finally, in [G96], [GM98], the authors try to extend face lifting to higher dimensions using another strategy. They restrict themselves to polyhedra which can be written as intersections of cylindrifications of two-dimensional (arbitrary) polygons. This way all the operations are performed on the two-dimensional projections of the polyhedron. There are obvious advantages and shortcomings of this approach compared to the grid-based one, and only time will tell their relative performances in practice.

Acknowledgment We thank Mark Greenstreet for introducing us to the face lifting concept, and for answering many technical questions. Part of this work was done while the second author was visiting Berkeley, benefiting from discussions with P. Varaiya, S. Sastry, C. Tomlin, G. Pappas and many others. At VERIMAG we are indebted to comments of E. Asarin, O. Bournez and P. Caspi on dynamical systems and to the help of Y. Raoul and S. Tripakis in software engineering.

References

- [AD94] R. Alur and D.L. Dill, A Theory of Timed Automata, *Theoretical Computer Science* 126, 183–235, 1994.
- [ACH⁺95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis and S. Yovine, The Algorithmic Analysis of Hybrid Systems, *Theoretical Computer Science* 138, 3–34, 1995.
- [AM95] E. Asarin and O. Maler, Achilles and the Tortoise Climbing Up the Arithmetical Hierarchy, in P.S. Thiagarajan (Ed.), *Proc. FST/TCS'95*, 471–483, LNCS 1026, Springer, 1995.
- [AMP95-a] A. Asarin, O. Maler and A. Pnueli, Reachability Analysis of Dynamical Systems having Piecewise-Constant Derivatives, *Theoretical Computer Science* 138, 35–66, 1995.

- [AMP95-b] E. Asarin, O. Maler and A. Pnueli, Symbolic Controller Synthesis for Discrete and Timed Systems, in P. Antsaklis, W. Kohn, A. Nerode and S. Sastry(Eds.), *Hybrid Systems II*, LNCS 999, Springer, 1995.
- [AC84] J.-P. Aubin and A. Cellina, *Differential Inclusions: Set-valued Maps and Viability Theory*, Springer, 1984.
- [C93] P. Caspi, Global Simulation via Partial Differential Equations, Unpublished note, Verimag, 1993.
- [G96] M.R. Greenstreet, Verifying Safety Properties of Differential Equations, in *Proc. CAV'96*, 277-287, 1996.
- [GM98] M.R. Greenstreet and I. Mitchell, Integrating Projections, these proceedings.
- [HW96] T.A. Henzinger and H. Wong-Toi, Linear Phase-Portrait Approximation for Nonlinear Hybrid Systems, in R. Alur, T.A. Henzinger and E.D. Sontag (Eds.), *Hybrid Systems III*, 377-388, LNCS 1066, Springer, 1996.
- [HKPV95] T.A. Henzinger, P.W. Kopke, A. Puri and P. Varaiya, What's Decidable about Hybrid Automata?, *Proc. 27th STOC*, 373-382, 1995.
- [HS74] M.W. Hirsch and S. Smale, *Differential Equations, Dynamical Systems and Linear Algebra*, Academic Press, 1974.
- [KM91] R.P. Kurshan and K.L. McMillan, Analysis of Digital Circuits Through Symbolic Reduction, *IEEE Trans. on Computer-Aided Design*, 10, 1350-1371, 1991.
- [LTS97] J. Lygeros, C. Tomlin and S. Sastry, Multiobjective Hybrid Controller Synthesis, in O. Maler (Ed.), *Proc. Int. Workshop on Hybrid and Real-Time Systems*, 109-123, LNCS 1201, Springer, 1997.
- [PBV96] A. Puri, V. Borkar and P. Varaiya, ϵ -Approximation of Differential Inclusions, in R. Alur, T.A. Henzinger and E.D. Sontag (Eds.), *Hybrid Systems III*, 363-376, LNCS 1066, Springer, 1996.
- [PV94] A. Puri and P. Varaiya, Decidability of Hybrid Systems with Rectangular Differential Inclusions, in D. Dill (Ed.), *Proc. CAV '94*, LNCS 1066, Springer, 1996.
- [RW89] P.J. Ramadge and W.M. Wonham, The Control of Discrete Event Systems, *Proc. of the IEEE* 77, 81-98, 1989.
- [S96] J.A. Sethian, *Level Set Methods : Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge, 1996.
- [SKE97] O. Stursberg, S. Kowalewski and S. Engell, Generating Timed Discrete Models of Continuous Systems, in *Proc. MATHMOD'97*, Vienna, 1997.
- [TPS98] C. Tomlin, G. Pappas, and S. Sastry, Conflict Resolution for Air Traffic Management: A Study in Multi-Agent Hybrid Systems, *IEEE Trans. on Automatic Control*, to appear.

Automotive Control Revisited

Linear Inequalities as Approximation of Reachable Sets

Ansgar Fehnker *

CSI

P.O. Box 9010, 6500 GL Nijmegen, the Netherlands
ansgar@cs.kun.nl

ABSTRACT: Reachability analysis of hybrid system imposes restrictions on the continuous and discrete behavior. In this paper a method is proposed to approximate the reachable set of linear systems by linear inequalities. It allows to use the full continuous dynamics of hybrid systems for reachability analysis. This method is applied to an automotive control problem, which was presented by Stauner et al. in [SMF97].

1 Introduction

This paper presents an approximation technique for reachable sets of hybrid systems and applies this technique to a problem known from literature. Stauner, Müller and Fuchs presented in [SMF97] an automotive control problem as real-life benchmark problem for the analysis of embedded reactive systems. They verified some safety-properties for a system which controls the height of one wheel of a car. They determined upper and lower bounds on the height, they showed that the (extended) controller does not change the height in bends, and proved that two special control locations can not be attained at the same moment. In addition Stauner et al. examined the step response (in the sense of Control Theory) of the system.

Verification of safety properties, which impose restrictions on the reachable states, requires the use of approximation techniques, because the exact reachable sets are difficult to compute and difficult to handle. In general there are two possibilities to cope with this problem. First, one can use an approximation of the hybrid system, i.e. specify a hybrid system with simpler continuous dynamics, which includes the behavior of the original system. Stauner et al. used an approximation of nonlinear hybrid systems by linear hybrid systems, i.e. systems where the continuous behavior is governed by variables with piecewise constant derivatives [SMF97, Sta97]. This method is based on the method presented in [HH95, HWT96]. The second possibility is to approximate the reachable sets, but to use (a slight approximation of) the full continuous dynamics of the original specification. Puri, Borkar and Varaiya presented an approximation technique for Lipschitz differential inclusions [PBV95] using a small perturbation of the original system. These perturbations use variables with piecewise con-

* Research supported by Netherlands Organization for Scientific Research (NWO) under contract SION 612-14-004

stant derivatives. The approximation technique presented in this paper is of the second type. It uses bounded polyhedra, which include the reachable sets, and requires that the continuous behavior is governed by piecewise linear differential equations.

The following section presents the HIOA model of Lynch et al. A short description of the automotive control problem is given in the third section. In section 4 and 5 some aspects of linear inequalities and linear systems are discussed, leading to an approximation method. The last section presents some results for the automotive control problem.

2 The HIOA model

We use the model of Hybrid I/O Automata (HIOA) by Lynch, Segala, Vaandrager and Weinberg [LSVW96] for the description of systems which show both continuous and discrete behavior. This model allows shared variables as well as shared actions. Within this model it is possible to reason about composition of hybrid systems, implementation relations between systems and it allows to describe the continuous behavior of hybrid systems separately from the discrete behavior.

A hybrid I/O automaton (HIOA) $\mathcal{A} = (U, X, Y, \Sigma^{in}, \Sigma^{int}, \Sigma^{out}, \Theta, \mathcal{D}, \mathcal{W})$ consists of:

- Three disjoint sets of *input*, *internal* and *output variables* U, X, Y , respectively. Let V be the union of these sets. \mathbf{V} is the set of valuations of V . Valuations will also be called *states*.
- Three disjoint sets $\Sigma^{in}, \Sigma^{int}, \Sigma^{out}$ of *input*, *internal* and *output actions*. Σ^{in} contains e , a special *environment* action, which models the occurrence of input which is unobservable except (possibly) through its effect on input variables. Σ denotes the union of the input, internal and output actions.
- A nonempty set Θ , a subset of \mathbf{V} , containing the *initial states*. This set is closed under change of values for input variables.
- A set $\mathcal{D} \subseteq \mathbf{V} \times \Sigma \times \mathbf{V}$ of *discrete transitions*. By definition each input action of a HIOA is always enabled. The environment action only affects inputs and the input variables may change, when a discrete transition occurs.
- A set of *trajectories* \mathcal{W} over V . A trajectory w is a mapping from I to states, where I is a left-closed interval of the *time axis* $\mathbb{R}^{\geq 0}$, with left endpoint equal to 0. (In general it is sufficient to define the time axis as subgroup of the real numbers with addition.) \mathcal{W} must contain point trajectories, it has to be closed under subintervals and if a trajectory w restricted to $[0, t]$ is an element of \mathcal{W} for all $t \in \mathbb{R}^{\geq 0}$, then w has to be an element of \mathcal{W} , too. We assume in this paper that w is integrable.

An important concept of HIOA is that of *hybrid executions*. A *hybrid execution fragment* α is an alternating infinite or finite sequence of trajectories and actions $\alpha = w_0 a_1 w_1 \dots$. If α is a finite sequence then it ends with a trajectory. We call α a *hybrid execution*, when the first state of α is an element of Θ . A state s is

defined to be *reachable* if there exists a finite hybrid execution, with last state equal to s .

The *hybrid trace* of an hybrid execution records the visible behavior of the execution. The set of all hybrid traces describes the external behavior of a HIOA. A HIOA \mathcal{A} *implements* a HIOA \mathcal{B} , if the traces of \mathcal{A} are a subset of the traces of \mathcal{B} . \mathcal{A} implements \mathcal{B} requires that \mathcal{A} and \mathcal{B} are *comparable*, meaning they have the same external actions and the same external variables. A *simulation relation* (or just simulation) is usually used to prove that the traces of HIOA \mathcal{A} are a subset of the traces of a HIOA \mathcal{B} . A simulation is a relation which maps all states of a hybrid execution α of \mathcal{A} to states of some hybrid execution of \mathcal{B} , such that the traces of these executions are the same. For more detail see [DL97] or [HSV94].

Complex hybrid systems can be modeled by *composing* HIOAs. Two HIOAs \mathcal{A} and \mathcal{B} can be composed if they are *compatible*, which means they have no output actions or output variables in common and no internal variable of either is a variable of the other. The composition of two compatible HIOA is itself an HIOA. The input variables of the composition are the union of \mathcal{A} and \mathcal{B} 's input variables minus the union of their output variables. The same holds for the input actions. A HIOA is *closed* if there are no input actions or input variables. Consequently the environment action has no influence on closed systems and can be omitted in the specification. Considering the automotive control problem, we will see in section 3 that the EHC and filtered environment are modeled with input and output, but the composition of these has no input at all.

Hybrid systems typically use two types of variables: variables which range over finite (or at most countable) sets, and variables which range over (a subset of) \mathbb{R} . The model of Alur et al [ACH⁺95] uses locations and data variables for this purpose. We define V_D as the set of discrete variables and V_C as the set variables ranging over reals. We can define V as $V_D \cup V_C$, and the set of valuations \mathbf{V} as $\mathbf{V}_D \times \mathbf{V}_C$. We identify \mathbf{V}_C with (a subset of) \mathbb{R}^n . Let s_D and s_C denote the projection of the state s on \mathbf{V}_D and \mathbf{V}_C respectively.

Transitions are specified in *precondition/effect* style (table 1 and 2). Preconditions are predicates with variables from V . If a transition is enabled and eventually taken, the state is changed according to the specification of the effect. If a precondition is *true* or the effect is defined by identity, it is usually omitted. When a transition takes place, the values of the input variables may change arbitrarily. We call a hybrid system *clocked* with sampling time t_{sample} , if discrete transitions may only occur every t_{sample} time units. See for example table 2.

3 The System

This section presents the hybrid system used for the automotive control problem [SMF97] in terms of the HIOA model by Lynch et al. The model used by Stauner et al. is followed as close as possible. For further technical details and a motivation of the specific choices within this model see [SMF97] or [Sta97].

The system consists of different components. First, we have the chassis, whose

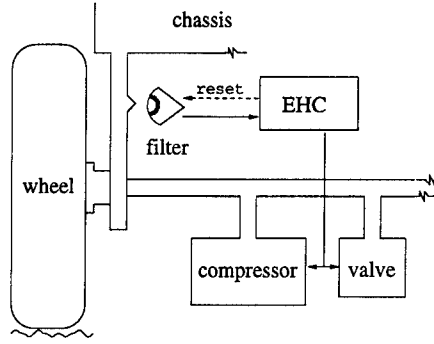


Fig. 1. The EHC in its environment

height can be changed by pneumatic suspension with a compressor and an escape valve. The height is measured by a low-pass filter, which filters disturbances of high frequency, caused for example by holes in the road. The electronic height control (EHC) uses the filtered height to decide, whether to use the compressor, the escape valve, or to do nothing.

The chassis level is influenced by external disturbances and by the escape valve and compressor. The rate of change of the height h of the chassis is the sum of the changes due to disturbances, denoted by e , and the changes due to compressor and escape valve, denoted by c . The continuous behavior of h is modeled by the linear differential equation

$$\dot{h} = e + c \quad (1)$$

If the controller uses the escape valve, the height h decreases with a rate c in the interval $[ev_{min}, ev_{max}]$, while using the compressor increases the height h with $c \in [cp_{min}, cp_{max}]$. The bounds of the disturbances are given by $e \in [e_{min}, e_{max}]$. To ensure that the EHC is able to avoid unbounded increase or decrease of height we assume $e_{min} = ev_{max}$ and $e_{max} = cp_{min}$. Of course, one would prefer more realistic and less restrictive assumptions as that the average influence of the environment has to be smaller than the average influence by the controller. Stauner et al. believed "(...) that the limits of the expressiveness of (linear) hybrid automata are reached with statements of this kind" [SMF97, p. 144].

The filter keeps track of the height, with the restriction that it takes some time until changes in height are properly detected. This feature is useful, because it limits the influence of short and small disturbances. The filter is modeled by

$$\dot{f} = \frac{1}{T}(h - f) \quad (2)$$

Here the constant T determines the time the filter needs to adjust the filtered height properly. The filter also has an input action **back** (synchronization label *set_f* in [SMF97]), which allows to reset the filtered height to the setpoint *sp*. The filtered environment (table 1) describes the behavior of the height and the

actions:	continuous variables:	init:
input: back	input: $c \in \mathbb{R}$	$e \in [e_{min}, e_{max}]$
internal: none	internal: $e, h \in \mathbb{R}$	$h = sp$
output: none	output: $f \in \mathbb{R}$	$f = sp$
discrete transitions:		
back:	Effect: $f := sp$	
trajectories: w is an I -trajectory, if the following holds for all $t \in I$:		
	$w.\dot{f} = \frac{1}{T}(h - f)$	
	$w.\dot{h} = e + c$	
	$w.e \in [e_{min}, e_{max}]$	

Table 1. The filtered environment

filtered height due to input of the EHC and disturbances by the environment.

Initially the controller is in control location `in_tolerance` and neither the escape valve nor the compressor are used, thus $c = 0$. If the filtered height exceeds an upper limit otu , then the controller enters control location `down`, with effect that the height decreases with a rate $c \in [ev_{min}, ev_{max}]$. If the controller is in location `down` and the filtered height gets smaller than a given upper limit itu , then the controller re-enters control location `in_tolerance` and resets the filtered height to the setpoint sp . Similarly there is a control location `up`, which is entered if the filtered height f falls below a lower limit otl , with effect $c \in [cp_{min}, cp_{max}]$. In this case the controller re-enters `in_tolerance` when f exceeds itl . To get a realistic model we assume $otl \leq itl \leq sp \leq itu \leq otu$.

The controller uses different values for otl, itl, itu, otu depending on whether the car is driving or stopped, denoted by indices d and s . If the controller leaves `in_tolerance` it makes a nondeterministic choice between the modes *driving* and *stopped*. The HIOA of the EHC (table 2)² uses the modes s for the stopped car and d for the driving car. The model assumes additionally that transitions can only be taken every t_{sample} seconds.

In the remainder of this paper matrix and vector multiplication are used. We assume that all matrices and vectors have elements in \mathbb{R} and are of a proper size. The block matrix $\begin{pmatrix} A \\ B \end{pmatrix}$ will be denoted as $(A; B)$. A^T and a^T denotes the transposition of the matrix A and vector a respectively. We assume a norm $\|\cdot\|$ like the Euclidean norm. The maximum, minimum, compactness of sets etc. are defined with respect to this norm.

4 Transitions and Linear Inequalities

The composition of the EHC and the filtered environment has some useful properties, which allow a reachability analysis of this system. The system is clocked, the enabling conditions of the transitions are defined by linear inequalities and

² The hybrid automata used in [SMF97] uses on some places strict equalities like $>$.

actions	continuous variables	discrete variables
input: none	input: $f \in \mathbb{R}$	input: none
internal: stay	internal: $t_{clock} \in \mathbb{R}^{\geq 0}$	internal: $mode \in \{d, s\}$
output: to_down, to_up, back	output: $c \in \mathbb{R}$	$loc \in \{\text{down}, \text{up}, \text{in_tolerance}\}$
init: $t_{clock} = 0 \wedge c = 0 \wedge loc = \text{in_tolerance}$		output: none
discrete transitions:		
to_down(m):		to_up(m):
Pre: $\wedge t_{clock} = t_{sample}$ $\wedge loc \in \{\text{in_tolerance}, \text{up}\}$ $\wedge (loc = \text{up}) \rightarrow (m = mode)$ $\wedge f \geq otu_m$		Pre: $\wedge t_{clock} = t_{sample}$ $\wedge loc \in \{\text{in_tolerance}, \text{down}\}$ $\wedge (loc = \text{down}) \rightarrow (m = mode)$ $\wedge f \leq otl_m$
Eff: $loc := \text{down}$ $t_{clock} := 0$ $c := [ev_{min}, ev_{max}]$ $mode := m$		Eff: $loc := \text{up}$ $t_{clock} := 0$ $c := [cp_{min}, cp_{max}]$ $mode := m$
stay:		back:
Pre: $\wedge t_{clock} = t_{sample}$ $\wedge \vee \wedge loc = \text{in_tolerance}$ $\wedge \vee f \in [otl_s, otu_s]$ $\vee f \in [otl_d, otu_d]$ $\vee \wedge loc = \text{down}$ $\wedge f \geq itu_{mode}$ $\vee \wedge loc = \text{up}$ $\wedge f \leq itl_{mode}$		Pre: $\wedge t_{clock} = t_{sample}$ $\wedge \vee \wedge loc = \text{down}$ $\wedge f \in [otl_{mode}, itu_{mode}]$ $\vee \wedge loc = \text{up}$ $\wedge f \in [itl_{mode}, otu_{mode}]$
Eff: $t_{clock} := 0$		Eff: $loc := \text{in_tolerance}$ $t_{clock} := 0$ $c := 0$
trajectories: w is an I -trajectory, if the following holds for all $t \in I$:		
$w.t_{clock} = 1$		
$w.t_{clock} \leq t_{sample}$		
If $w.loc = \text{in_tolerance}$		then $w.c = 0$
If $w.loc = \text{up}$		then $w.c \in [cp_{min}, cp_{max}]$
If $w.loc = \text{down}$		then $w.c \in [ev_{min}, ev_{max}]$

Table 2. The EHC

the assignments are linear. Additionally the continuous behavior is governed by piecewise linear differential equations and the initial set is a bounded polyhedron. The composition of the EHC with the filtered environment is also closed. The main components of a hybrid system are the transitions and the trajectories, sometimes referred to as discrete transitions and continuous transitions. In this and the next section we discuss some features of both (discrete) transitions and trajectories.

Many examples of hybrid systems use linear inequalities for the specification

of transitions or to define the set of initial states. Linear inequalities occur also in approximation techniques of nonlinear hybrid systems [HH95, PBV95] and are also used to verify invariants hybrid regular expressions [XHT97].

Given the linear inequality

$$a^T x \leq b \quad (3)$$

with $x \in \mathbb{R}^n$, $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$, the set of solutions K is a half-space of \mathbb{R}^n . The vector a is a *normal* on the hyper-plane which separates K from K^c , i.e. a is orthogonal to the hyper-plane, and points to the complement of K . An intersection of halfspaces is called *polyhedron*. Using the matrix product we can define a polyhedron as set of solutions $K \subseteq \mathbb{R}^n$ of

$$Ax \leq b \quad (4)$$

A is a $m \times n$ matrix, with m the number of inequalities, $b \in \mathbb{R}^m$ is a vector and ' \leq ' means that each element of Ax is less than or equal to the corresponding element of b .

We see that the EHC and the filtered environment have preconditions and effects of a special structure. The atomic predicates over continuous variables are of the form $A s_C \leq b$, with $s_C, b \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$. Consequently the preconditions of the EHC define unions respectively intersections of polyhedra. Note that no strict inequality like ' $<$ ' is allowed, because for computational reasons we want that all the polyhedra are closed, i.e. they contain their boundaries (see footnote 2). Given a HIOA \mathcal{A} which uses strict inequalities, we can replace all ' $<$ ' by ' \leq ' and we yield a HIOA \mathcal{B} which is implemented by \mathcal{A} .

Polyhedra are *convex*, which means that if x_1 and x_2 are elements of a polyhedron, the *convex combination* $\lambda x_1 + (1-\lambda)x_2 \in K$ is an element of the polyhedron for all $\lambda \in [0, 1]$. The *convex hull* of set K , $\text{conv}(K)$, is the smallest polyhedron which contains all convex combinations of points $x_1, x_2 \in K$. Given a finite set $\{p_1, \dots, p_k\}$ of points in \mathbb{R}^n , the convex hull of this set is a bounded polyhedron, also called a *polytope*. The *vertices* of this polytope will be members of the set $\{p_1, \dots, p_k\}$, though not every point p_i needs to be a vertex.

In the next section it will be necessary to find a maximum of a linear function with linear constraints. This problem is often referred to as LP-problem (*Linear Programming*). There are several equivalent forms for the LP-problem, but it is usually defined as follows: Given $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$ find the maximum

$$\max\{c^T x \mid Ax \leq b\} \quad (5)$$

If $K = \{x \mid Ax \leq b\}$ is a nonempty polytope, it is possible to calculate the vertices p_1, \dots, p_k of K . This leads to the following lemma:

Lemma 1. *Let $K = \text{conv}(p_1, \dots, p_k)$ be a polytope and suppose $c : \mathbb{R}^{\geq 0} \rightarrow \mathbb{R}^n$ such that each element c_i is analytic. Then there exist $t_{\max} > 0$ and a vertex p*

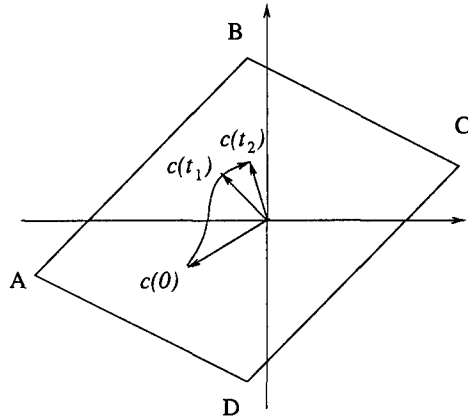


Fig. 2. $c(t)$ moves from $c(0)$ to $c(t_2)$

of K which satisfies

$$c(t)^T p = \max_{x \in K} c(t)^T x \quad \forall t \in [0, t_{max}] \quad (6)$$

This lemma says that there exist a vertex which is optimal at $t = 0$ and which stays optimal for at least t_{max} time units. The optimum does not have to be unique, and if two or more vertices are optimal then all points which are a convex combination of these points are also optimal. The proof uses the fact that all functions $c(t)^T p_i$ are analytic mappings from \mathbb{R} to \mathbb{R} . Among these there has to be a function which is greater than or equal to the other on an interval. More information on polyhedra and solving the LP-problem can be found in [Sch86], [GMW91] or [Fis91].

Figure 2 illustrates lemma 1 for two dimensions. The maximum (6) is attained from 0 up to t_1 in vertex A. At time t_1 the maximum is not unique, it is attained in every point on edge AB. Without the assumption $c(t)$ analytic, one could easily construct a function $c(t)$ (e.g. using $\sin(1/t)$), such that the maximum will not be constantly attained in one vertex, for any interval $[0, t_{max}]$.

Because polyhedra are convex the theorems on convex sets, or even stronger theorems, are holding. A nonempty intersection of two polyhedra is a polyhedron, and a nonempty intersection of a polytope with a polyhedron yields a polytope. The image of a linear function of a (bounded) polyhedron is also a (bounded) polyhedron.

The effect of the transitions of the EHC is specified by an assignment $s'_D := \phi_D$, where ϕ_D is a mapping from V_D to V_D and an assignment $s'_C := A s_C + b$, where A is $m \times n$ matrix and $b \in \mathbb{R}^n$. We will also allow nondeterministic assignments to polytopes. We already have seen that the preconditions define sets of polyhedra and with the foregoing we can conclude that the effect will map these to a set of polyhedra.

5 Trajectories and Linear Systems

In the last section we have seen that polyhedra are important in studying transitions. This section presents a method which uses polyhedra as approximation of reachable sets. Therefore we need some theory on linear time invariant systems. For a general introduction to Control Theory and Linear systems see [BG80] or [Bro70].

Considering the composition of EHC with the filtered environment we see that the behavior of the continuous variables is modeled in two ways. We divide the set of continuous variables V_C in two sets Z and Q . The trajectories of the variables in Z are continuous functions $z : I \rightarrow \mathbf{Z}$, which are defined by differential equations. Q contains the variables that take only values in bounded subsets of \mathbb{R} . Though Q is a subset of V_C , the trajectories of these variables are generally not continuous,

We assume that the continuous behavior of the variables in Z is defined by a linear, time invariant differential equation as follows

$$\dot{z}(t) = \mathbf{A} z(t) + \mathbf{B} q(t) \quad (7)$$

In the remainder of this section we will call z (internal) state and q input, for both are defined analogous to states and inputs as known in Control Theory, even if the corresponding variables are not input or internal variables of the HIOA. The filtered environment (table 1) uses $Z = \{f, h\}$ and $Q = \{e, c\}$.

The unique solution of the differential equation (7) with initial state $z(0) = z_0$ is given by ³

$$z(t) = e^{\mathbf{A}t} z_0 + \int_0^t e^{\mathbf{A}(t-\sigma)} \mathbf{B} q(\sigma) d\sigma \quad (8)$$

We abbreviate the right-hand side by $\varphi(z_0, t, q)$.

In Control Theory one often wants to find a time optimal control for the system (7), assuming $q : I \rightarrow \mathbf{Q}$ and $z_0 \in \mathbf{Z}_0$, with \mathbf{Q} and \mathbf{Z}_0 bounded and closed subsets of \mathbb{R}^m and \mathbb{R}^n respectively. Let $Reach(\mathbf{Z}_0, t_f, \mathbf{Q})$ denote the set of states that can be reached from the initial set of states \mathbf{Z}_0 at time t_f with inputs in \mathbf{Q} . Denote the boundary of a set S by $\delta(S)$.

The reachable states form a convex, bounded set. So, if we assume $z_f \in \delta(Reach(\mathbf{Z}_0, t_f, \mathbf{Q}))$ then there exists a supporting hyperplane (tangent plane) that contains z_f . Let c_f be the normal on this hyperplane such that $c_f^T z \leq c_f^T z_f$ for all reachable states z . However, if we assume that the reachability set is unknown, we can not choose such a z_f and determine the normal c_f . But fortunately it is possible to find for a given c_f and an arbitrary t_f , an input \bar{q} and an initial state z_0 such that $c_f^T z \leq c_f^T \varphi(z_0, t_f, \bar{q})$ for all $z \in Reach(\mathbf{Z}_0, t_f, \mathbf{Q})$.

Lemma 2. Suppose $\mathbf{Z}_0 \subseteq \mathbb{R}^n$ and $\mathbf{Q} \subseteq \mathbb{R}^m$ are compact and convex sets. Let c_f be a vector in \mathbb{R}^n and $t_f \in I$. Then there exists a $\bar{z}_0 \in \delta(\mathbf{Z}_0)$ and a mapping

³ Note that $e^{\mathbf{A}t}$ is a symbolic notation for the fundamental solution of $\dot{z} = \mathbf{A} z$

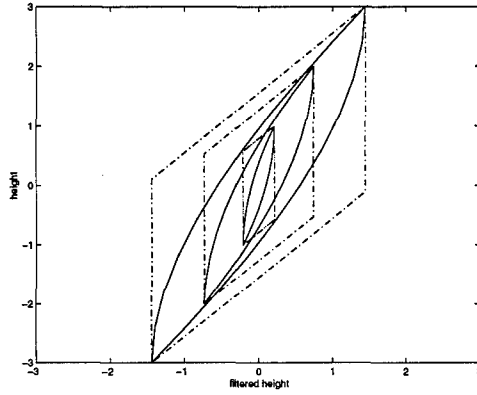


Fig. 3. The bounds on reachable height and filtered height after 1, 2 and 3 seconds (solid) and their approximations (dash-dotted). Initially $f = 0$, $h = 0$ and $loc = in_tolerance$.

$\bar{q} : I \rightarrow \delta(Q)$ with

$$c_0^T \bar{z}_0 = \max_{z_0 \in Z_0} c_0^T z_0 \quad (9)$$

$$c_0^T e^{-A^T t} \bar{q}(t) = \max_{q \in Q} c_0^T e^{-A^T t} q, \quad \forall t \in [0, t_f] \quad (10)$$

$$c_f^T \bar{z}_f = \max_{z_f \in Z_f} c_f^T z_f \quad (11)$$

with $\bar{z}_f = \varphi(\bar{z}_0, t_f, \bar{q})$, $c_0 = e^{A^T t_f} c_f$ and $Z_f = Reach(Z_0, t_f, Q)$.

(9) and (10) can be proved using the fact that there always exists a maximum of a linear function on a compact and convex set (see [LM67]). It should be noted that \bar{z}_0 and \bar{q} are not necessarily unique. Using (9), (10) and (8) shows straightforward that $c_f^T (\varphi(\bar{z}_0, t_f, \bar{q}) - \varphi(z_0, t_f, q)) \geq 0$ holds for arbitrary z_0 and q , therefore (11) is proven. The relations between z_f , u and z_0 given by this lemma, are used to prove the bang-bang principle (or theorem of Lee-Markus [LM67]). This principle states that it is always possible to reach an extreme state with an extreme control.

Suppose we want to find for a given matrix $A = (c_1^T; \dots; c_l^T)$ a vector $b = (b_1; \dots; b_l)$ such that $Reach(Z_0, t_f, Q) \subseteq \{z | Az \leq b\}$. Let $c_{0i} := e^{A^T t_f} c_i$ then we can find \bar{z}_0 , from (9) and an optimal \bar{q}_i from (10). Then $c_i^T z \leq b_i$ holds for all elements z in the reachability set, with $b_i := c_i^T \varphi(\bar{z}_0, t_f, \bar{q}_i)$. If we apply this method to all c_i , we get that the reachable set is included in the polyhedron defined by $Az \leq b$. Whether the polyhedron is bounded depends on the choice of matrix A . Figure 3 shows an example of how the reachable states of the composition of the EHC and the filtered environment were approximated (using the same matrix A as in the next section). We can approximate the reachable

states arbitrarily close by adding proper row-vectors to A .

Often we are not only interested in the reachable set on certain points in time, but also in constraints on the reachable states in an interval of time. To get a lemma similar to 2, we need restrictions on \mathbf{Z}_0 and \mathbf{Q} . In the remainder of this section we assume that both \mathbf{Z}_0 and \mathbf{Q} are not only convex and compact but also bounded polyhedra. $\text{Reach}(\mathbf{Z}_0, [0, t_f], \mathbf{Q})$ denotes the set of states which can be reached from \mathbf{Z}_0 with input in \mathbf{Q} within time t_f .

Lemma 3. Suppose $\mathbf{Z}_0 \subseteq \mathbb{R}^n$ and $\mathbf{Q} \subseteq \mathbb{R}^m$ are bounded polyhedra. Let c_{max} be a vector in \mathbb{R}^n . Then there exists a $\bar{z}_0 \in \delta(\mathbf{Z}_0)$, and a $t_{max} > 0$ and a constant $\bar{q} \in \{q | q : [0, t_{max}] \rightarrow \delta(\mathbf{Q})\}$ with

$$c_{max}^T e^{A t} \bar{z}_0 = \max_{z_0 \in \mathbf{Z}_0} c_{max}^T e^{A t} z_0, \quad \forall t \in [0, t_{max}] \quad (12)$$

$$c_{max}^T e^{A(t_{max}-t)} \bar{q}(t) = \max_{q \in \mathbf{Q}} c_{max}^T e^{A(t_{max}-t)} q, \quad \forall t \in [0, t_{max}] \quad (13)$$

$$c_{max}^T \bar{z}(t) = \max_{z(t) \in \mathbf{Z}(t)} c_{max}^T z(t), \quad \forall t \in [0, t_{max}] \quad (14)$$

with $\bar{z}(t) = \varphi(\bar{z}_0, t, \bar{q})$ and $\mathbf{Z}(t) = \text{Reach}(\mathbf{Z}_0, t, \mathbf{Q})$.

The proof uses the existence of an interval $[0, t_{max1}]$ on which the maximum of $c_{max}^T e^{A t} z_0$ is attained in one vertex \bar{z}_0 of \mathbf{Z}_0 (see lemma 1). The same holds for \bar{q} ; there exists an interval $[0, t_{max2}]$ where \bar{q} is constant. Take t_{max} as minimum of t_{max1} and t_{max2} . Similar to the proof of lemma 2 we use (12), (13) and (8) to show that $c_{max}^T (\varphi(\bar{z}_0, t, \bar{q}) - \varphi(z_0, t, q)) \geq 0$ holds for arbitrary z_0, q and all $t \in [0, t_{max}]$, and therefore (14) is proven. For \bar{q} is constant on $[0, t_{max}]$ we are able to simplify the integrals that arise from (8)

Suppose \mathbf{Z}_0 is a bounded polyhedron, and matrix A is chosen such that \mathbf{Z}_0 is contained in the bounded polyhedron $\{z | A z \leq b_0\}$ for a certain b_0 . Then lemma 3 allows to find a $b(t) := c_{max}^T \bar{z}(t)$ such that $\text{Reach}(\mathbf{Z}_0, t, \mathbf{Q}) \subseteq \{z | A z \leq b(t)\}$ for all t in some interval $[0, t_{max}]$. The upper bound t_{max} depends solely on the matrix A , \mathbf{Q} and \mathbf{Z}_0 and not on the choice of b_0 . This allows to approximate the reachable set even if $t_{sample} > t_{max}$. In this case the approximation technique is applied iteratively to the result of the preceding approximation, until the number of iterations times t_{max} exceeds t_{sample} .

We are often also interested in a single bounded polyhedron that includes the states which are reachable within interval $[0, t_{max}]$. For this purpose we can use the inequality $A z \leq \max_{t \in [0, t_{max}]} b(t)$. In general this approximation gets worse with a longer interval $[0, t_{max}]$. It should be noted that $\{(z; t) | z \in \text{Reach}(\mathbf{Z}_0, t, \mathbf{Q})\}$ is generally not convex, and hence it is difficult to handle transitions that do not take place at a specified time. Therefore we restrict the model to clocked HIOAs.

Lemma 3 provides a method to approximate the reachable set of such a HIOA. Let $\text{REACH}(t, \Theta)$ denote the set of reachable states at time t with initial set Θ and let $\text{REACH}([0, t], \Theta)$ denote the set of all states, which are reachable within time t . The approximation of the continuous parts of the reachable states with bounded polyhedra of the form $\{z | A z \leq b\}$, is denoted by

$\text{aprx}(\text{REACH}(\cdot, \Theta), A)$. Let S be a set of states, then $\text{trans}(S)$ is the set of states after applying the transition rules.

Let Θ denote the set of initial states. We choose a matrix A and a vector b such that the initial values of the variables in Z are included in bounded polyhedra of the form $\{z | Az \leq b\}$. The algorithm used for the reachability analysis has the following structure. Initially we choose $\Theta_0 = \Theta$ and $i = 0$. In the first step we determine the bounds of $\text{aprx}(\text{REACH}([0, t_{\text{sample}}], \Theta_i), A)$. In the next step we determine $\theta_i := \text{aprx}(\text{REACH}(t_{\text{sample}}, \Theta_i), A)$ and $\Theta_{i+1} := \text{trans}(\theta_i)$. We increase i by one and return to the first step. The algorithm terminates if $\theta_{i+1} \subseteq \theta_i$. Notice that there are no guarantees that the algorithm terminates and it is easy to find a counterexample.

We can apply this algorithm when the hybrid system is clocked, its continuous behavior is governed by piecewise linear differential equations, the transitions are defined by linear inequalities and the assignments are linear. In the previous section we have seen that the composition of the EHC and the filtered environment has these properties.

6 Results

In [SMF97] Stauner et al. use HYTECH to verify properties of the composed system. First, they showed that the EHC keeps the height of the chassis within certain bounds. Next, they proved that the escape valve and compressor are never used at the same time. They had to include two automata, which model the escape valve and the compressor. In addition, they extended the model with a bend detection and showed that the EHC does not change the height in bends. Stauner et al. also examined the stability of the EHC after a step-like disturbance.

This paper re-examines only the bounds of the chassis level and the step response of the EHC. The second and third property involve mainly discrete variables. Consequently one can not expect that a different approximation technique improves these results. We used MATHEMATICA and MATLAB to re-examine the automotive control problem.

6.1 Bounds for the Chassis Level

The bounds of the chassis level are given by the maximum and minimum value of h of all reachable states. We derived the bounds of the chassis level for a system with $cp_{\min} = 1 \frac{\text{mm}}{\text{s}}$, $cp_{\max} = 2 \frac{\text{mm}}{\text{s}}$, $ev_{\min} = -2 \frac{\text{mm}}{\text{s}}$, $ev_{\max} = -1 \frac{\text{mm}}{\text{s}}$ and $sp = 0 \text{ mm}$. Therefore e lies in the interval $[-1 \frac{\text{mm}}{\text{s}}, 1 \frac{\text{mm}}{\text{s}}]$. The constants that define the transitions are chosen as $otl_s = -40 \text{ mm}$, $otu_s = 20 \text{ mm}$, $otl_d = -10 \text{ mm}$, $otu_d = 10 \text{ mm}$, $itl_s = -6 \text{ mm}$, $itus = 16 \text{ mm}$, $itl_d = -6 \text{ mm}$, $itu_d = 6 \text{ mm}$. Stauner et al. used as time constant of the filter $T = 2 \text{ s}$ and as sampling time $t_{\text{sample}} = 1 \text{ s}$. They chose $h = 0 \text{ mm}$ and $f = 0 \text{ mm}$ as initial values. Using this setting they verified that the chassis level h is always in $[-47 \text{ mm}, 27 \text{ mm}]$. This means that the outer limits otl_s and otu_s are never exceeded by more than 7 mm .

	$t_{sample} = 1\ s$	$t_{sample} = 0.5\ s$
$T = 2\ s$	$[-43.0, 23.6]$	$[-42.6, 23.3]$
$T = 1\ s$	$[-42.0, 22.1]$	$[-41.5, 21.6]$

Table 3. The bounds of the chassis level h in mm

They expected that the results can be improved by using a smaller time constant T and a smaller sampling time.

The behavior of f en h is governed by the following differential equation

$$\begin{aligned}\dot{f} &= \frac{1}{T}(h - f) \\ \dot{h} &= e + c\end{aligned}\tag{15}$$

with $e \in [-1 \frac{mm}{s}, 1 \frac{mm}{s}]$ and $c = 0 \frac{mm}{s}$, $c \in [-2 \frac{mm}{s}, -1 \frac{mm}{s}]$ or $c \in [1 \frac{mm}{s}, 2 \frac{mm}{s}]$ depending on the control location. With a more heuristic approach – using the solution of (15) – it can be found that the real height will exceed the outer tolerance limits with about $e_{max}(T + t_{sample})\ mm$ and $e_{min}(T + t_{sample})\ mm$ respectively.

We re-examine these results for $T \in \{2\ s, 1\ s\}$ and $t_{sample} \in \{0.5\ s, 1\ s\}$. We are mostly interested in the bounds on the height, but also in the bounds on the filtered height and the bounds on the difference between filtered and real height i.e. we want to determine the upper limits of f , $-f$, $f - h$, $h - f$, h and $-h$. Hence we use the matrix $A := (1\ 0; -1\ 0; 1\ -1; -1\ 1; 0\ 1; 0\ -1)$ to define the polyhedra that include the reachable filtered height and real height. Thus t_{max} from lemma 3 is equal to 1.

The initial states are $(f, h)^T \in \{x \in \mathbb{R}^2 | Ax \leq (10\ 30\ 20\ 20\ 10\ 30)^T\}$. This shortens the required time to run the algorithm. The influence of this choice on the results is limited, for all initial sets that contain the origin converge rapidly to the same collection of reachable sets. The results can be found in table 3. Obviously the proposed approximation technique gives better results. They coincide almost with the limits a more heuristic approach yields. In addition, the bounds are tighter for smaller T and t_{sample} as expected. It is worth to mention that states which satisfy $(loc = up) \wedge (f \leq sp + otl_{s/d})$ or $(loc = down) \wedge (f \geq sp + otu_{s/d})$ are not reachable within this setting. So the transitions `to_up` and `to_down` can be simplified.

6.2 Step Response of the EHC

In the previous subsection we examined the response of the EHC to small disturbances. In this subsection we assume that there only is one disturbance of step shape and no other disturbances occur. Disturbances of step shape are typical test functions to examine the stability of a controller. At an arbitrary moment, when the system is in `tolerance` and $f = 0\ mm$, the height makes a jump to j . This is the only disturbance and we assume $e = 0 \frac{mm}{s}$. The jump can be simulated by taking $f = 0\ mm$, $h = j$ and $t_{clock} \in [0\ s, 1\ s]$ as initial values. For the

j	[11,12]	[12,13]	[13,14]	[14,15]	[15,16]	[16,17]	[17,18]	[18,19]	[19,20]
t_{lmax}	5.8	4.6	4.0	3.6	3.2	3.0	2.8	2.7	2.5
t_{remax}	13.8	13.6	14.0	14.6	15.2	16.0	16.8	17.7	18.5
h_{end}	[0.1,4.4]	[0,4.4]	[0,4.3]	[0,4.4]	[0,4.3]	[0,4.3]	[0,4.2]	[0,4.3]	[0,4.2]

Table 4. Some results for the step response of the EHC. The values of t_{lmax} and t_{remax} are given in seconds, and those of j and h_{end} in mm .

quality of the approximations decrease with a longer interval, we took initially $t_{clock} \in [0\text{ s}, 0.1\text{ s}]$, $\dots t_{clock} \in [0.9\text{ s}, 1\text{ s}]$.

Stauner et al. assumed a jump $j \in (16\text{ mm}, 18\text{ mm}]$ and that the system is in *driving* mode. In this case we expect that the controller uses the escape valve after disturbances of this size. Additionally they assumed that the escape valve operates at its minimum value, hence $ev_{min} = ev_{max} = 1 \frac{mm}{s}$. This restriction was necessary to avoid arithmetic overflows. For this setting they found that the controller leaves `in_tolerance` at most 4.3 s after the disturbance and re-enters it after at most 22.3 s . They verified that the chassis level then lies in $[-1\text{ mm}, 6\text{ mm}]$.

For the same setting it is possible with the proposed approximation technique to show that the controller leaves `in_tolerance` at most after 3 seconds, re-enters `in_tolerance` at most after 16.9 seconds and the chassis level then lies in the interval $[3.0\text{ mm}, 4.1\text{ mm}]$. It is also possible to investigate the step response for a system with the original setting for ev_{min} and ev_{max} and a number of intervals for j . The results can be found in table 4. It shows for jumps j within the specified intervals the maximum time the controller needs to detect the disturbance (t_{lmax}), the maximum time the controller needs before re-entering `in_tolerance` (t_{remax}) and the interval that contains h after re-entering `in_tolerance` (h_{end}). The interval $[10\text{ mm}, 11\text{ mm}]$ was not considered, because a jump with $j = 10\text{ mm}$ can not be detected in finite time.

Figure 4 illustrates the behavior of the EHC due to jumps in $[13\text{ mm}, 14\text{ mm}]$ occurring at $t_{clock} = 0$. We see that the EHC enters location `down` after 3 seconds and re-enters location `in_tolerance` at least after 13 seconds. All reachable state will ultimately converge to points on the diagonal, for the filtered height converges to the real height.

7 Conclusion

The program used for the analysis is quite preliminary and it will cost some time to make it suitable for other problems than the automotive control problem. Hence the required CPU-times are less impressive (re-examining the step responses (table 4) lasted less than one hour, the reachability analysis required, depending on the choice for t_{sample} and T up, to one day). An extension to, or implementation into a more general program will be done in the future. To

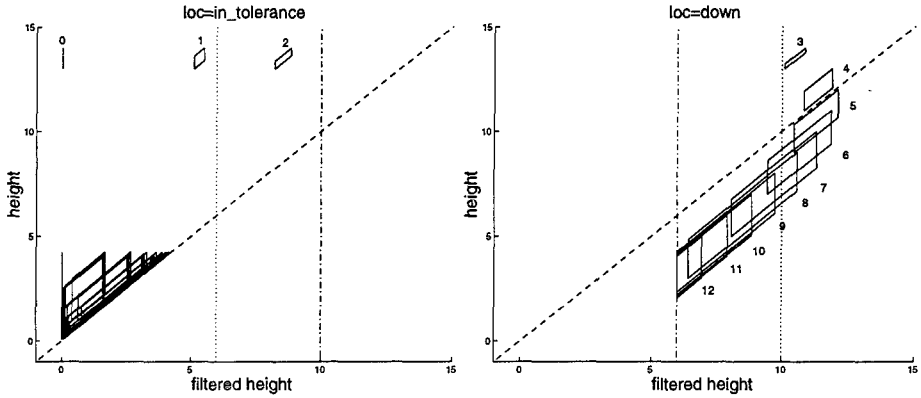


Fig. 4. Step response due to a jump in $[13 \text{ mm}, 14 \text{ mm}]$ starting at $t_{\text{clock}} = 0$

ensure that the algorithm is applicable to more realistic models, an analysis of the complexity will also be done in the future

The proposed approximation technique gives obviously better results for the reachability analysis of a hybrid system with restricted discrete dynamics, but uses the full dynamics of the linear system (linear in the sense of Control Theory) that describes the continuous dynamics. The most restrictive assumption is that the system has to be clocked. Fortunately many real-life problems within a controller-environment setting have this property.

If we apply our technique to systems with piecewise constant derivatives, this restriction can be weakened. The matrix \mathbf{A} in (7) will be zero, consequently t_{max} from lemma 3 will be infinity. Additionally the reachable sets will form a convex set in time and the approximation will be exact. This simplifies analysis of the system and hence the assumption of a clocked hybrid system can be dropped. For this class of hybrid system, the method proposed in this paper coincides with the methods for Linear Hybrid Systems as presented e.g. in [ACH⁺95].

Puri et al. approximated a Lipschitz differential inclusion arbitrarily close by a piecewise constant inclusion and then used polyhedra to approximate the reachable set [PBV95]. Lipschitz differential inclusions form a more general class of systems than linear systems. For in this paper only the latter were considered, it was not necessary to approximate the continuous behavior, assumed that the discrete behavior satisfies certain restrictions.

Reachability analysis of hybrid systems requires restrictions on the discrete and continuous dynamics. The hybrid system Stauner et al. used for the automotive control problem provides a discrete behavior that allows us to use properties of linear systems for analysis of the continuous part.

References

- [ACH⁺95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.
- [BG80] David Burghes and Alexander Graham. *Introduction to Control Theory, including Optimal Control*. Ellis Horwood series in mathematics and its applications. John Wiley & Sons, New York, 1980.
- [Bro70] Roger W. Brockett. *Finite Dimensional Linear Systems*. John Wiley & Sons, New York, 1970.
- [DL97] E. Dolginova and N. Lynch. Safety verification for automated platoon maneuvers: A case study. In Oded Maler, editor, *HART'97*, LNCS 1201. Springer-Verlag, 1997.
- [Fis91] Gerd Fischer. *Analytische Geometrie*. Friedr. Vieweg & Sohn, Braunschweig, 1991.
- [GMW91] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Numerical Linear Algebra and Optimization*, volume 1. Addison-Wesley, 1991.
- [HH95] T.A. Henzinger and P.-H. Ho. Algorithmic analysis of nonlinear hybrid systems. In P. Wolper, editor, *CAV 95: Computer-aided Verification*, LNCS 939, pages 225–238. Springer-Verlag, 1995.
- [HSV94] L. Helmink, M.P.A. Sellink, and F.W. Vaandrager. Proof-checking a data link protocol. In H. Barendregt and T. Nipkow, editors, *Proceedings International Workshop TYPES'93*, Nijmegen, The Netherlands, May 1993, LNCS 806, pages 127–165. Springer-Verlag, 1994. Full version available as CWI technical report.
- [HWT96] T.A. Henzinger and H. Wong-Toi. Linear phase-portrait approximations for nonlinear hybrid systems. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, LNCS 1066, pages 377–388. Springer-Verlag, 1996.
- [LM67] E.B. Lee and L. Markus. *Foundations of optimal control theory*. The SIAM series in applied mathematics. Wiley, New York, 1967.
- [LSVW96] N. Lynch, R. Segala, F.W. Vaandrager, and H.B. Weinberg. Hybrid I/O automata. In R. Alur, T. Henzinger, and E. Sontag, editors, *Hybrid Systems III*, LNCS 1066, pages 496–510. Springer-Verlag, 1996.
- [PBV95] A. Puri, V. Borkar, and P. Varaiya. ϵ -approximation of differential inclusions. In *Proceedings of the 34th IEEE Conference on Decision and Control (CDC 95)*, 1995.
- [Sch86] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1986.
- [SMF97] Thomas Stauner, Olaf Müller, and Max Fuchs. Using hytech to verify an automotive control system. In Oded Maler, editor, *HART'97*, LNCS 1201, pages 139–153. Springer-Verlag, 1997.
- [Sta97] Thomas Stauner. *Specification and Verification of an Electronic Height Control System using Hybrid Automata*. Master's thesis, Munich University of Technology, 1997.
- [XHT97] Li Xuandong, Dang Van Hung, and Zheng Tao. Checking hybrid automata for linear duration invariants. In *ASIAN' 97*, LNCS 1345, pages 166–180. Springer-Verlag, 1997.

Switching Controllers Based on Neural Network Estimates of Stability Regions and Controller Performance

Enrique D. Ferreira and Bruce H. Krogh

Department of Electrical and Computer Engineering
Carnegie Mellon University

email: edf/krogh@ece.cmu.edu

Abstract. This paper presents new results on switching control using neural networks. Given a set of candidate controllers, a pair of neural networks is trained to identify the stability region and estimate the closed-loop performance for each controller. The neural network outputs are used in the on-line switching rule to select the controller output to be applied to the system during each control period. The paper presents architectures and training procedures for the neural networks and sufficient conditions for stability of the closed-loop system using the proposed switching strategy. The neural-network-based switching strategy is applied to generate the switching strategy embedded in the SIMPLEX architecture, a real-time infrastructure for soft on-line control system upgrades. Results are shown for the real-time level control of a submerged vessel.

1 Introduction

A common approach to control complex dynamic systems is to design a number of different controllers, each for a particular operating region or performance objective, and then to switch among the controllers in real time to achieve the overall control objective. This is, for example, the philosophy behind gain-scheduled controllers. Recently, switching control strategies have been proposed for adaptive control of unknown systems [1],[9], and to optimize the performance of stabilizing controllers for a known plant [8].

It is useful to view switching control systems as hybrid systems, that is, systems with both continuous state variables and discrete state variables. The plant state variables (assuming a continuous-variable system) and possibly continuous state variables in the controllers constitute the continuous state of the switching control system; the index of the current controller being applied to the system, and possibly discrete variables in the sequential switching logic, constitute the discrete state. Sys-

tem performance and stability are also normally defined in terms of the continuous-state trajectories. Analysis of switching control systems from either perspective is difficult because of the interaction between the continuous and discrete dynamics through the switching rules.

Given a collection of controllers for a nonlinear dynamic system, neural network techniques are presented for estimating the regions of stability and performance of each controller, and an on-line switching strategy is proposed based on these neural network estimates. Sufficient conditions are presented for closed-loop stability of the switching control system. On the application side, we describe the use of the neural network strategy to implement the switching rules in the SIMPLEX architecture, a real-time environment developed at the Software Engineering Institute at Carnegie Mellon University that provides protection against errors in control system upgrades [11],[4] and [10]. Results are presented for the real-time control of the level of a submerged vessel.

2 Problem Formulation

We consider the problem of controlling a nonlinear system described by the state equations

$$\begin{aligned} \mathbf{x}_{k+1} &= f(\mathbf{x}_k, \mathbf{u}_k) \\ \mathbf{x}_k &\in S, \quad \mathbf{u}_k \in D \end{aligned} \tag{1}$$

where $\mathbf{x}_k \in R^n$ is the state vector and $\mathbf{u}_k \in R^m$ the control input vector. The connected sets $S \subset R^n, D \subset R^m$ represent physical constraints on the system state and control, respectively. The discrete-time state equation reflects the sampled-data implementation of a computer control system. The control objective is to take the state to the origin.

We assume M state feedback controllers have been designed for this system, with the i^{th} control law given by $g^i : R^n \rightarrow R^m$. We assume the origin is a stable equilibrium for each of the controllers in some (unknown) region. The objective of the switching strategy is to select one of the control outputs to apply the system at each control instant to achieve the largest possible region of stability for the closed-loop system with a good transient response.

The closed-loop system created by the application of each controller is characterized by a stability region and a performance index. The region of stability for controller i is defined as

$$R^i = \{ \mathbf{x}_o : \mathbf{x}_k^i(\mathbf{x}_o) \in S, \mathbf{g}^i(\mathbf{x}_k^i(\mathbf{x}_o)) \in D \quad \forall k \geq k_o \text{ and } \lim_{k \rightarrow \infty} \mathbf{x}_k^i(\mathbf{x}_o) = 0 \} \quad (2)$$

where $\mathbf{x}_k^i(\mathbf{x}_o)$ denotes the trajectory of the system (1) with the initial state \mathbf{x}_o at $k = 0$ under control law \mathbf{g}^i . We consider performance indices for the controllers of the form

$$J_\delta^i(\mathbf{x}_o) = B^i + \sum_{k=0}^{\infty} \delta^k U^i(\mathbf{x}_k^i(\mathbf{x}_o)), \quad i = 1, \dots, M \quad (3)$$

where $0 < \delta \leq 1$ is a discount factor, $U^i : R^n \rightarrow R$ is a positive definite state cost function, and B^i is the bias coefficient for the i^{th} controller. We assume (3) converges for $\delta = 1$.

The proposed approach for selecting the controller at each sampling instant is illustrated in figure 1. Neural networks are used to compute estimates of the stability regions R^i and performance indices J_δ^i at the current state, denoted by \hat{R}^i and \hat{J}_δ^i , respectively. The index of the control input to be applied for the next period, denoted i_k , is then selected as

$$i_k = \arg \min_{i \in I(\mathbf{x}_k, L_k)} \{ \hat{J}_\delta^i(\mathbf{x}_k) \} \quad (4)$$

where

$$I(\mathbf{x}_k, L_k) = \{ i | \mathbf{x}_k \in \hat{R}^i, \text{ and if } i \neq i_{k-1}, \hat{J}_\delta^i(\mathbf{x}_k) < L_k^i \}$$

with $L_k = \{ L_k^1, \dots, L_k^M \}$ and for $i = 1, \dots, M$

$$L_k^i = \begin{cases} \infty & \text{if } k = 0 \text{ or } \mathbf{x}_k \notin \hat{R}^{i_{k-1}} \\ L_{k-1}^i & \text{if } i \neq i_k \\ \min \{ \hat{J}_\delta^i(\mathbf{x}_k), L_{k-1}^i \} & \text{if } i = i_k \end{cases}$$

In words, the scheduler selects the controller with the minimum estimate performance index from among the controllers for which the current state is in the estimated stability region and, for the controllers other than the current controller, the current estimated performance index is less than the corresponding bound L_k^i . The limits L_k^i guarantee a controller is not re-selected once it has been used until its performance index has decreased below the lowest value reached when it was last active. If the system leaves the stability region of the controller used during the previous period, all controllers become candidates again by setting $L_k^i = \infty$ for all $i = 1, \dots, M$. This selection criterion is motivated by the *min*-

switching strategies based on Lyapunov functions proposed in [8] and [6]. The strategy proposed in this paper replaces the Lyapunov functions with the neural network estimates of the performance measure, making it viable for systems for which the dynamics are not known precisely or Lyapunov functions cannot be found by analysis.

3 Neural Networks

Neural networks are used in two ways in the proposed scheme. The *stability region estimator* is a classifier, identifying when the system is stable for a given state. The *performance estimator* produces an estimate of the cost-to-go function (3) from a given state. In both cases a two-layer feed-forward network is used for its capacity as an universal approximator with a size that is small relative to the size of the data set [3].

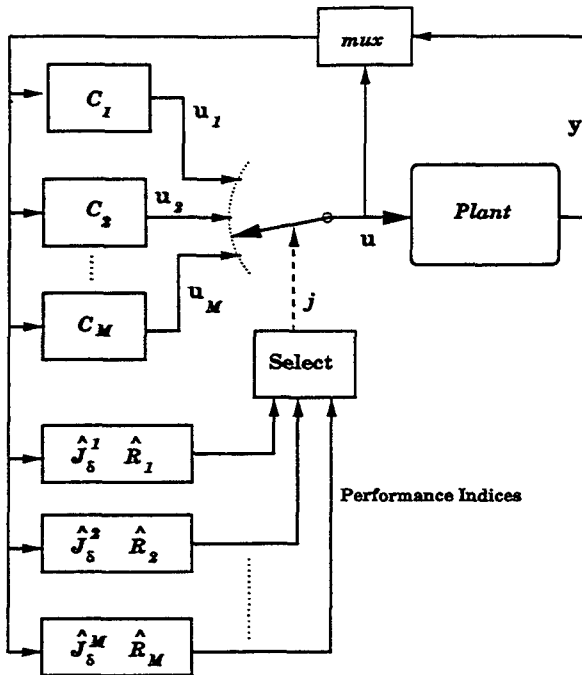


Fig. 1. Control Scheduling diagram.

The input-output behavior of the two-layer network with linear outputs units is described by

$$\mathbf{y} = \mathbf{b}_o + W_o^T \mathbf{x} + W_3^T \phi_2(\mathbf{b}_2 + W_2^T \phi_1(\mathbf{b}_1 + W_1^T \mathbf{x})), \quad (5)$$

where $\mathbf{x} \in \mathcal{R}^{n_o}$ is the input vector, $\mathbf{y} \in \mathcal{R}^{n_3}$ the output vector, $W_i \in \mathcal{R}^{n_i \times n_{i-1}}$, $i = 1, 2, 3$ and $W_o \in \mathcal{R}^{n_3 \times n_o}$, the weight matrices for each layer of n_i units, $\mathbf{b}_i \in \mathcal{R}^{n_i}$ the threshold vectors for each layer and $\phi_i: \mathcal{R}^{n_i} \rightarrow \mathcal{R}^{n_i}$, $i = 1, 2$; the nonlinear functions for each hidden layer. The functions $\phi_i(\cdot)$ for the hidden units of the neural network are all chosen to be the hyperbolic tangent functions applied to each component of its input vector (i.e. $\phi_{ij}(\mathbf{x}) = \tanh(x_j)$). In our application, the input vector \mathbf{x} is the state of the plant for both the stability region estimators and performance estimators.

3.1 Estimating Stability Regions

For the stability region estimators, the neural network output \mathbf{y} is a two-dimensional vector with components ranging roughly between -1 and 1 in the region of approximation. The ideal output values are $(y_1, y_2) = (1, -1)$ when \mathbf{x} belongs to R^i and $(y_1, y_2) = (-1, 1)$ when \mathbf{x} is not in R^i . To make a distinct classification in the non-ideal case (i.e., when the components of \mathbf{y} are not equal to ± 1), two positive threshold parameters, θ and δ , are selected to implement the following decision rule:

Stability Region Classifier: Declare \mathbf{x} belongs to R^i if and only if:

1. $y_1(\mathbf{x}) - y_2(\mathbf{x}) > \theta$, and
2. $\|\nabla_{\mathbf{x}}(y_1(\mathbf{x}) - y_2(\mathbf{x}))\| < \delta$,

where the notation $\nabla_{\mathbf{x}}$ denotes the gradient with respect to \mathbf{x} .

The stability region classifier is motivated by the necessity of obtaining conservative approximations for the stability regions. The parameter θ is chosen after the network is trained so that the classification is correct for all the training and validation data. The parameter δ is chosen much smaller than the maximum of the norm of the gradient of the network output over the domain.

The training of the neural network for the stability region estimator is based on supervised learning procedures. This approach is widely used for pattern recognition and classification applications [3]. To initialize the training for each controller, the following three regions $A \subset B \subset C$ are defined, based on a priori knowledge of the closed-loop system behavior:

1. *Inner region A.* A very conservative region which includes all the states from which convergence to the origin is certain.
2. *Study region B.* The region on which the training procedure is going to be conducted.
3. *Unsafe region C.* The bounding region in which the system is either unstable or the state is outside the operating region of interest.

By making experiments with the controllers starting at states belonging to region B, and observing if the evolution leads the system to region A or to region C, data is obtained for region B to train the neural network. This procedure is carried out initially using off-line data, but training can continue on line as the system operates.

Comparisons of the neural network stability region estimator with other approaches to stability region approximation have been presented in [7]. We have found that in all cases, with a reasonable amount of training, the neural network obtains an estimate of the stability region which is much less conservative than most other methods. Moreover, since it is not model-based, the neural network classifier can be applied to systems using empirical data.

3.2 Estimating Performance Indices

Estimating performance indices such as (3) is a standard problem in Neuro-dynamic programming [2]. A Heuristic Dynamic Programming (HDP) algorithm [13] is used to train the networks. The training algorithm uses an estimation of the cost-to-go at \mathbf{x}_k given by

$$J_{\delta}^{i*}(\mathbf{x}_k) = U^i(\mathbf{x}_k) + \delta \hat{J}_{\delta}^i(\mathbf{x}_{k+1}) \quad (6)$$

where $J_{\delta}^{i*}(\mathbf{x}_k)$ is the desired value for the network for state \mathbf{x}_k . Equation (6) is motivated by the definition of $J_{\delta}^i(\mathbf{x})$ (3) neglecting the bias term B^i which is added directly to the output of the network. In our applications, $U^i(\mathbf{x})$ is a standard quadratic form,

$$U^i(\mathbf{x}) = \mathbf{x}^T P \mathbf{x}, \quad P^T = P > 0, \quad i = 1, \dots, M.$$

The HDP training procedure, illustrated in figure 2, is described briefly as follows. Given the new state value \mathbf{x}_{k+1} at time $k+1$, the neural network with parameters from time k , denoted $NN_i(k)$, is used to predict both $\hat{J}_{\delta}^i(\mathbf{x}_k)$ and $\hat{J}_{\delta}^i(\mathbf{x}_{k+1})$. The latter value is used to compute $J_{\delta}^{i*}(\mathbf{x}_k)$ as defined in (6). The difference $\epsilon_k = J_{\delta}^{i*}(\mathbf{x}_k) - \hat{J}_{\delta}^i(\mathbf{x}_k)$ is used, in a back-

propagation algorithm, to update the parameters in the neural network to produce $NN_i(k+1)$ (indicated by the arrow through the NN_i block).

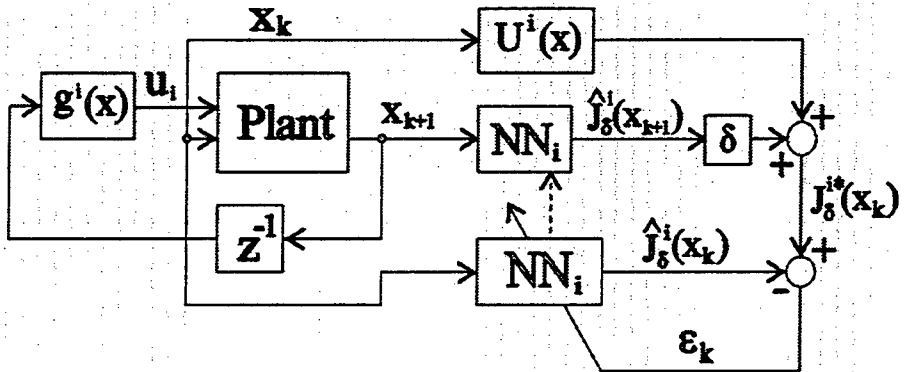


Fig. 2. HDP learning scheme.

Analytical results for related problems in the context of Q-learning [5] and temporal differences [12] indicate that convergence should be expected under rather mild conditions. A principal difference between our application and most work on learning cost-to-go performance indices is that the estimated values of the performance indices do not influence the control laws. We assume, rather, that each of the given controllers stabilizes the system and the feedback laws remain fixed.

4 Analysis of Closed-Loop Performance

We first consider the system behavior in the perfect information case, that is, when the performance measures and stability regions are known for each controller. We then consider the effect of using the neural network estimators rather than the exact values.

The approach to analyzing the closed-loop system follows the technique for the min-switching strategy suggested in [8] for the continuous-time case. To restate the basic Lyapunov results for our discrete-time context, suppose for each control law g^i there is a known Lyapunov function $V^i(x)$ for the closed-loop system under that control law within the region of stability R^i . Moreover, suppose the control applied at each sample instant is chosen according to the min-switching strategy, that is, the control is selected which corresponds to a Lyapunov function with the minimum value among all the Lyapunov functions evaluated at the current state. The following theorem is the discrete-time version of the result in [8].

Theorem 1. *If the system given by (1) is controlled by the min-switching strategy applied to a set of known Lyapunov functions, the origin is asymptotically stable in the region $R = \bigcup R^i$. Moreover, the function*

$$W(x) = \min_{i \in \{j \mid x \in R^j\}} \{V^i(x)\} \quad (7)$$

is a Lyapunov function on R .

Proof. Follows from the continuous-time result in [8], *mutatis mutandis*.

We now apply this result to the min-switching strategy considered in this paper by observing that if the performance indices J_1^i (3) converge ($\delta = 1$), they are in fact Lyapunov functions for the respective controllers.

Theorem 2. *Given the system defined by (1) and a collection of control laws $g^i, i = 1, \dots, M$. Suppose for each control law the origin is asymptotically stable for the closed-loop system*

$$x_{k+1} = F^i(x_k)$$

in a connected region R^i , and J_δ^i given by (3) converges for $\delta = 1$. Then there exists a $\delta^ \in (0, 1)$ such that the origin is asymptotically stable in the region $R = \bigcup R^i$ for the closed-loop system controlled by the min-switching strategy [8] for any $\delta \in (\delta^*, 1]$. Moreover, for any $\delta \in (\delta^*, 1]$ the function*

$$J_\delta = \min_{i \in \{j \mid x \in R^j\}} \{J_\delta^i\} \quad (8)$$

is a Lyapunov function on R .

Proof. For each i , if J_δ^i converges, it follows from the definition of J_δ^i that it is continuous in δ and therefore there exists some $\delta_i^* \in (0, 1)$ such that for all $\delta \in (\delta_i^*, 1]$ J_δ^i is a Lyapunov function for the closed-loop system under control law i on R^i . The theorem follows by letting $\delta^* = \max(\delta_1^*, \dots, \delta_M^*)$.

We now turn to the min-switching strategy using the neural network estimators. In the following we assume the stability region estimators

are all conservative, that is, for all $i = 1, \dots, M$, $\hat{R}^i \subseteq R^i$. Moreover, we assume all the stability region estimates are nonempty and connected. These assumptions are reasonable given the properties of neural network classifiers and the ability to initiate the training for the stability region estimators based on a priori knowledge of the capabilities of the given controllers.

Theorem 3. *Suppose the assumptions of Theorem 2 are satisfied and the performance estimates are computed using $\delta \in (\delta^*, 1]$ where δ^* is as specified in Theorem 2. Furthermore, suppose J_δ^i is continuous on R^i . If for some $\epsilon > 0$ the performance estimates satisfy*

$$|\hat{J}_\delta^i(x) - J_\delta^i(x)| < \epsilon \quad \text{for all } x \in \hat{R}^i, i = 1, \dots, M \quad (9)$$

and the min-switching strategy is applied for some $x_0 \in \hat{R} = \bigcup \hat{R}^i$ resulting in a state trajectory such that there exists $K \in \mathcal{N}$ for which $x_k \in \bigcap \hat{R}^i, \forall k > K$, then $x_k \rightarrow X_\epsilon$ where

$$X_\epsilon = \bigcup_i X_\epsilon^i = \bigcup_i \{x \mid J_\delta^i(x) \leq \sup_{\bar{x} \in \hat{R}^i} J_\delta^i(\bar{x})\} \quad (10)$$

and

$$X_\epsilon^i = \{x \in R^i \mid -\Delta J_\delta^i(x) \leq 2\epsilon\}.$$

Proof. For a given $x_0 \in \hat{R}$, let i_k be the sequence of controllers selected by the min-switching rule. If there exists some K and $l \in \{1, \dots, M\}$ such that $i_k = l$ for all $k > K$, the theorem is true since the origin is a stable equilibrium for controller l . On the other hand, if the controller switches infinitely often, there must be one controller l which is selected infinitely often. Let the sequence of time indices $0 < k_1 < k_2 \dots$ be an infinite sequence of sampling instants when controller l is selected with $x_k \in \bigcap \hat{R}^i, \forall k > k_1$. The min-switching rule implies

$$\hat{J}_\delta^l(x_{k_{j+1}}) < \hat{J}_\delta^l(x_{k_j}), \quad \forall j$$

because of the limits L_{k_j} . Since the $\hat{J}_\delta^l(x_{k_j})$ are bounded from below, the sequence $\hat{J}_\delta^l(x_{k_j})$ converges to some constant C .

For each $i = 1, \dots, M$ let $\eta^i(x)$ denote the error in the i^{th} performance estimate at state x where it is assumed $|\eta^i(x)| \leq \epsilon$ for some $\epsilon > 0$. This implies that when the i^{th} controller is applied at a state $x \in \hat{R}^i$,

$$\begin{aligned}
\Delta \hat{J}_\delta^i(\mathbf{x}) &\triangleq \hat{J}_\delta^i(F^i(\mathbf{x})) - \hat{J}_\delta^i(\mathbf{x}) \\
&= \Delta J_\delta^i(\mathbf{x}) + \eta^i(F^i(\mathbf{x})) - \eta^i(\mathbf{x}) \\
&\leq \Delta J_\delta^i(\mathbf{x}) + 2\epsilon.
\end{aligned}$$

Since J_δ^i is a Lyapunov function for the system under controller i , $\Delta J_\delta^i(\mathbf{x}) \leq 0$. Returning to the specific controller l , suppose that there are an infinite number of the \mathbf{x}_{k_j} that remain a finite distance from the set

$$\chi_\epsilon^l = \{\mathbf{x} \in R^l \mid -\Delta J_\delta^l(\mathbf{x}) \leq 2\epsilon\}.$$

This would imply the sequence $\Delta \hat{J}_\delta^l(\mathbf{x}_{k_j})$ is negative and bounded away from zero infinitely often, contradicting $\hat{J}_\delta^l(\mathbf{x}_{k_j}) \rightarrow C$. Therefore, $\mathbf{x}_{k_j} \rightarrow X_\epsilon^l$. More precisely, given any $\tilde{\epsilon} > 0$, there exists some K_ϵ^l such that the distance $d(\mathbf{x}_{k_j}, \chi_\epsilon^l) < \tilde{\epsilon}$ for all $k_j > K_\epsilon^l$. This is illustrated in figure 3.

While controller l is applied, $J_\delta^l(\mathbf{x}_k)$ is monotone non-increasing since J_δ^l is a Lyapunov function for the system. Therefore, $J_\delta^l(\mathbf{x}_k) \leq J_\delta^l(\mathbf{x}_{k_j})$, for $k \geq k_j$ until another controller becomes active (see figure 3). Define $I_l = \{k \in \mathcal{N} \mid i_k = l\}$ and $\bar{J}_\delta^l = \sup_{\tilde{\mathbf{x}} \in X_\epsilon^l} J_\delta^l(\tilde{\mathbf{x}})$. Then, for any given $\beta > 0$ we

have

$$J_\delta^l(\mathbf{x}_k) \leq \sup_{k_j} J_\delta^l(\mathbf{x}_{k_j}) \leq \bar{J}_\delta^l + \beta \quad \forall k \in I_l, k > K_\epsilon^l$$

because of the continuity assumption on $J_\delta^l(\mathbf{x})$. Since this has to be true for any l , after a finite K in which all the controllers that are not used infinite number of times do not become active anymore, the sequence \mathbf{x}_k is arbitrarily close to X_ϵ .

This theorem indicates that when the performance estimates are used rather than the exact performance indices, the min-cost switching strategy will drive the state to a neighborhood of the origin determined by the magnitudes of the errors in the performance estimates for each controller. Theorem 3 does not guaranteed the neighborhood is arbitrarily small, however. Moreover, the exact performance measures are not known in general, so the neighborhood in Theorem 3 could not be computed even if the bound on the estimation error was known. These difficulties are eliminated when $\delta = 1$, however, since in this case we have $\Delta J_\delta^i(\mathbf{x}) = -U(\mathbf{x})$.

Corollary 4. Under the assumptions of Theorem 3 with $\delta = 1$, if the min-switching strategy is applied for some $x_o \in \hat{R} = \bigcup \hat{R}^i$ resulting in a state trajectory such that $x_k \rightarrow \bigcap \hat{R}^i$, then

$$x_k \rightarrow \{x \in R^n | U(x) \leq 2\epsilon\}.$$

5 An Application

One of the principal motivations for developing the controller switching strategy presented in this paper is to provide a method for implementing the switching rules in the SIMPLEX architecture, a real-time environment developed at the Software Engineering Institute at Carnegie Mellon University that provides protection against errors in control system upgrades. Figure 4 shows a typical configuration for SIMPLEX in which there are three controllers: a *safety* controller, a reliable *baseline* controller, and

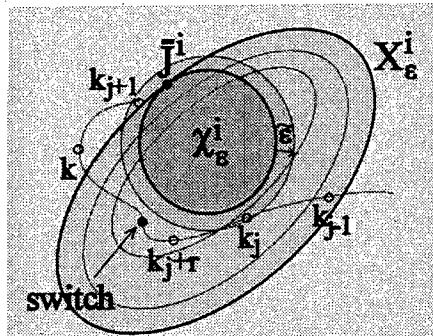


Fig. 3. Trajectory of the system illustrating convergence to X_ϵ .

an *experimental* controller representing a new, untested control module. The basic idea of the SIMPLEX system is to guarantee that the baseline controller performance is maintained if there are problems in the experimental controller. This is accomplished by monitoring the control outputs and system performance when the experimental controller is installed, and switching control back to the baseline controller if problems are detected. The safety controller is invoked when it is necessary to take more extreme action to return the system to the operating region for the baseline controller.

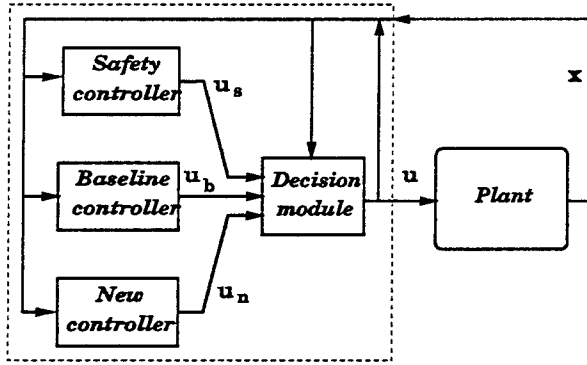


Fig. 4. SIMPLEX architecture.

Clearly the ability for the SIMPLEX system to provide the desired protection against errors in the experimental controller depends entirely on the rules used to switch between controllers. These rules are very difficult to create and maintain, even for small systems. The neural network approach proposed in this paper provides a means of obtaining less conservative estimates of the stability regions for the controllers, and also a method for determining when to switch from the safety controller back to the baseline controller based on estimates of their performance.

We present results here on the implementation of the *min-switching* control strategy for a level-control system for an underwater vessel. This system has been designed in the Software Engineering Institute at Carnegie Mellon University as a testbed for the development of dependable and evolvable systems using the SIMPLEX architecture. The experimental system consists of a water tank in which a vessel can move vertically by changing the size of the air bubble inside it. Air is moved in and out of the vessel through a flexible tube connected to a cylinder-piston mechanism. Figure 5 shows a schematic diagram of the system components. The control goal is to stabilize the vessel at an arbitrary position inside the water tank. The position of the vessel and the size of the air bubble are measured directly using ultrasound sensors. A stepper motor controls the piston movement. Constraints are imposed by the bottom of the tank and the water level. The control input is limited by the maximum speed of the stepper motor.

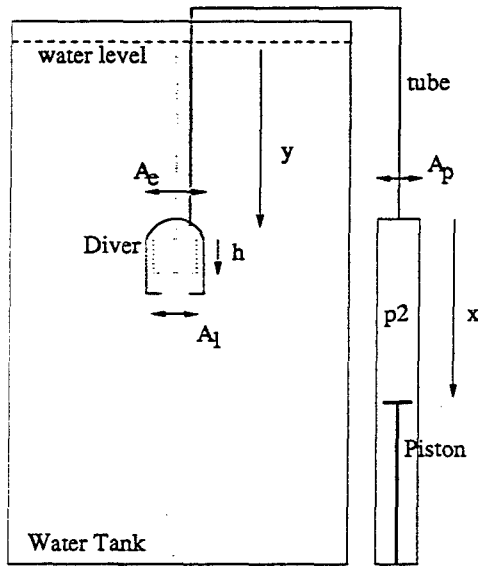


Fig. 5. Schematic diagram for the submerged vessel system.

A set point of $y_{st} = 25$ in was selected. Two controllers were used to test the switching strategy. Both controllers are state feedback controllers used in the original SIMPLEX architecture implementation. One controller, u_1 , has an acceptable performance close to the set point while the second controller, u_2 , performs better in a larger operating region using a bang-bang action, but with unacceptable oscillations near the set point. An analytical model was used for initial training of the neural network, then experimental data from twenty runs were used to adapt the parameters of the neural networks to estimate the performance indices of both controllers. Figure 6 shows a typical data profile to estimate the performance index of one of the controllers and figure 7 shows a slice of the resulting performance estimate.

Figure 8 shows a switching experiment for a step change in the setpoint value for y_{st} from 13 to 25 inches. Figure 9 shows the estimated performance indices during the run. From the figure we observe that controller 2 is preferred for larger values of y . After approximately 5.5 seconds \hat{J}_g^1 becomes smaller than \hat{J}_g^2 and the scheduler switches to controller 1.

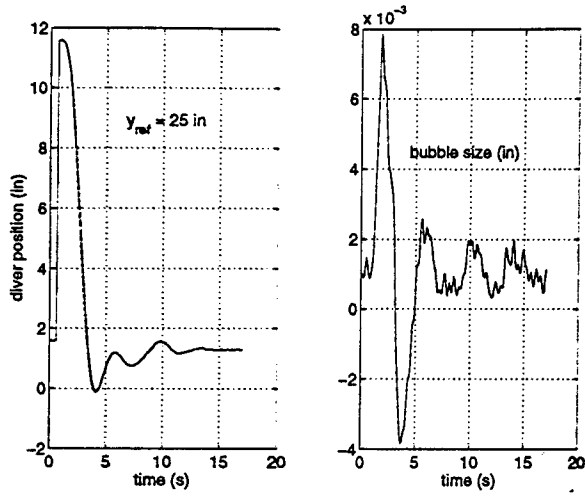


Fig. 6. Data obtained from an experimental run for a controller.

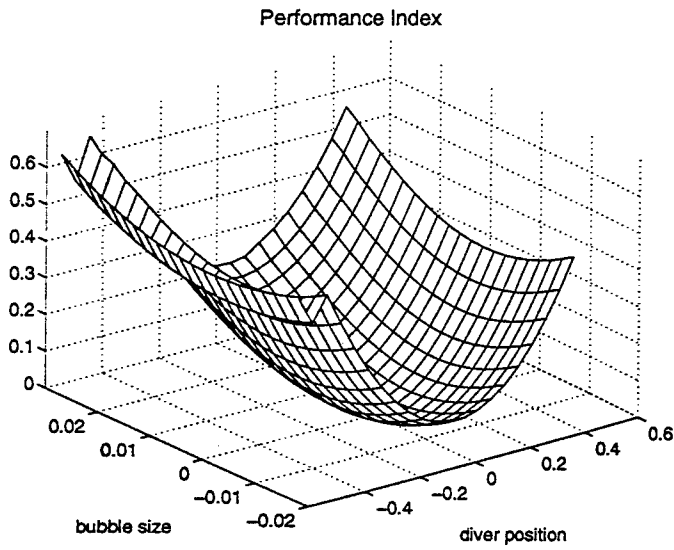


Fig. 7. Performance index estimate ($\dot{y} = 0$).

6 Discussion

This paper presents a method for switching among a set of given controllers using multilayer feedforward neural networks and neuro-dynamic programming techniques. In contrast to switching control strategies aimed

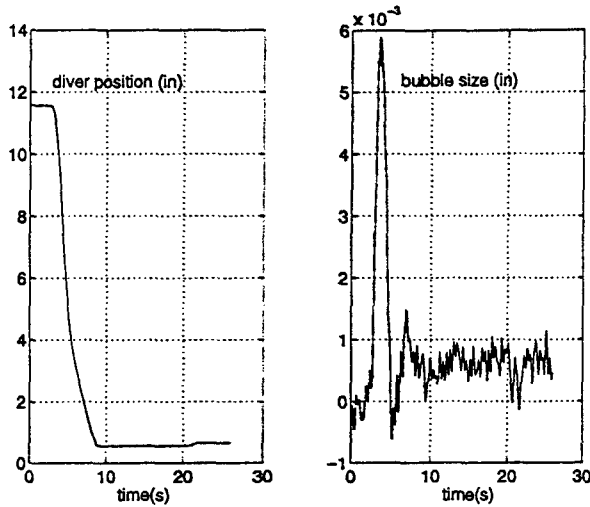


Fig. 8. Level relative position of the vessel and bubble size during a switching experiment.

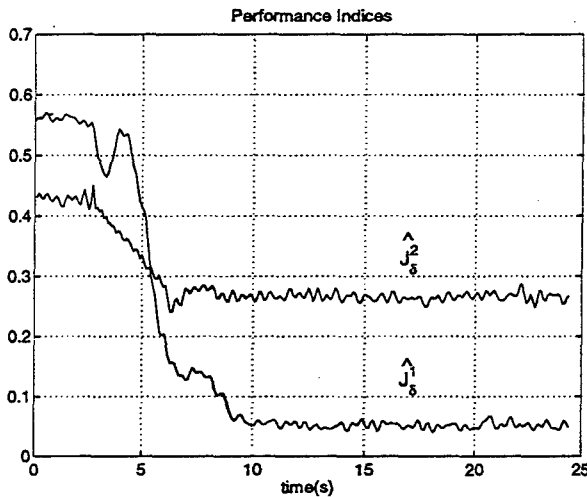


Fig. 9. Performance indices estimates for controllers for the submerged vessel during a switching experiment.

at adapting to unknown plant dynamics, the objective in this work is to select the best controller from among a set of controllers that have been designed for a known plant. This objective is most closely aligned with the switching control strategies proposed in [8] and [6]. By using neural networks to estimate the stability regions and performance indices for the controllers, the switching strategy depends on experimental data from the actual system, rather than analytical models that may lead to misleading or incorrect switching rules. We present a new result on the stability of the min-switching strategy using estimates of Lyapunov functions.

The convergence and stability results in this paper are sufficient conditions. There are several open problems concerning the verification of these conditions in applications and the possibility of obtaining less conservative results. For the closed-loop behavior, the ramifications of continued learning and persistent excitation need to be studied further. It would be desirable to introduce techniques by which performance estimate learning could be achieved for the controllers that are not currently controlling the system, by introducing, perhaps, a model of the system being controlled. The introduction of adaptive control to deal with changes in the plant dynamics may also be important for some applications.

Acknowledgments

This research has been supported by the Uruguayan government through a CONICYT-IBD grant, the Organization of American States and DARPA, contract number F33615-97-C-1012.

References

1. J. Balakrishnan and K.S. Narendra. Adaptive control using multiple models. *IEEE Trans. on Automatic Control*, 42(2):171-187, Feb 1997.
2. Dimitri P. Bertsekas and John Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.
3. Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, Great Britain, 1995.

4. M. Bodson, J. Lehoczky, R. Rajkumar, L. Sha, D. Soh, M. Smith, and J. Stephan. Control configuration in the presence of software failures. In *Proc. 32nd IEEE Conf. Decision Control*, volume 3, page 2284, San Antonio, TX, Dec 1993.
5. S.J. Bradtke, B.E. Ydstie, and A.G. Barto. Adaptive linear quadratic control using policy iteration. In *Proc. American Control Conference*, volume 3, pages 3475–9, Baltimore, MD, Jun 1994.
6. Michael S. Branicky. Stability of switched and hybrid systems. In *Proc. 33rd IEEE Conf. Decision Control*, volume 4, pages 3498–3503, Lake Buena Vista, FL, Dec 1994.
7. E.D. Ferreira and B.H. Krogh. Using neural networks to estimate regions of stability. In *Proc. of 1997 American Control Conference*, volume 3, pages 1989–93, Albuquerque, NM, Jun 1997.
8. J. Malmberg, B. Berhardsson, and K.J. Aström. A stabilizing switching scheme for multi-controller systems. In *Proc. of the IFAC World Congress*, volume F, pages 229–234, San Francisco, California, USA, 1996. IFAC'96, Elsevier Science.
9. A.S. Morse. Supervisory control of families of linear set-point controllers - part i: Exact matching. *IEEE Trans. on Automatic Control*, 41(10):1413–31, Oct 1996.
10. D. Seto, L. Sha, A. Chutinan, and B.H. Krogh. The simplex architecture for safe on-line control system upgrades. Submitted to 1998 American Control Conference.
11. L. Sha. A software architecture for dependable and evolvable industrial computing systems. In *Proc. IPC'95*, pages 145–156, Detroit, MI, May 1995.
12. J.N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Trans. on Automatic Control*, 42(5):674–690, May 1997.
13. P. Werbos. A menu of designs in reinforcement learning over time. In W.T. Miller III, R.S. Sutton, and P. Werbos, editors, *Neural Networks for control*, chapter 3. MIT Press, 2nd edition, 1991.

A Logic for the Specification of Continuous Systems*

Viktor Friesen

Technische Universität Berlin
Forschungsgruppe Softwaretechnik (Sekt. FR5-6)
Franklinstr. 28/29
D-10587 Berlin, Germany
e-mail: friesen@cs.tu-berlin.de

Abstract. The paper proposes a first-order logic for the specification of continuous components of hybrid systems. The particularity of the approach lies in its interpretation of individual variables not as functions over time or as point-based values, but as environment-based values. An environment-based value closely models the local behavior of a function defined on a continuous time domain. The advantage of the approach is that it enables us to consider the derivation operator as an ordinary unary logical function. Thus, the logic is free from any built-in operators; they can all be defined on the elements of the carrier set of environment-based values.

To facilitate the definition of additional logical functions and predicates like limit, derivation of arbitrary order or continuity, the user is allowed to specify them in the intuitive notation of functions defined on time. The semantics of the logic provides two lifting operators, which lift the functions and the predicates to the appropriate semantic spaces. These lifting operators do not violate the intuitive meaning of the introduced constructs. An outline of the proof of this fact is given.

1 Introduction

A specification of a system usually determines a set of permitted system behaviors. If T is the considered time domain¹, V a finite set of system variables, and Val a set of possible values, then a *state* is an assignment of a value to each system variable and a *behavior* is an assignment of a state to each time point. The set of all states is denoted by Σ and the set of all system behaviors by BEH , i.e. $\Sigma := V \rightarrow Val$ and $BEH := T \rightarrow \Sigma$. With these definitions, the task of a specification is to define an appropriate subset of BEH .

To formally specify hybrid systems, powerful specialized description techniques for discrete and continuous parts of the system are needed. Also required

* This work is being funded as part of the KONDISK program of the German Research Foundation (DFG).

¹ T is often chosen as a finite or infinite interval of real numbers \mathbb{R} or as the natural numbers \mathbb{N} .

are means for combining discrete and continuous components naturally. For more than twenty years, numerous logics and model-based languages have been available, which are tailored to the proper specification of discrete systems like VDM [7] or Z [11]. The central concepts on which these languages are based are *state invariants* and *operations*. The system is specified by describing a state invariant $INV \subseteq \Sigma$ and a finite set of operations o_1, \dots, o_n , each of them denoting a binary relation on Σ . Roughly speaking, the invariant specifies the static, and the operations the dynamic, aspects of a system. The system variables are interpreted as elements of Val . The state invariant INV and the operations o_i are specified using predicates on Σ and $\Sigma \times \Sigma$, respectively, both formulated in predicate logic. In the case of real-time extensions of such formalisms, the user can additionally describe some time aspects of the system behavior, like the duration of an operation or the time the system remains in a particular state. Because ordinary predicate logic is used, the available toolkit can be extended by defining supplementary functions and predicates, thus leading to flexibility in the choice of the specification means, cf. [7, 11].

Continuous systems cannot be described in the same style, since their dynamics is not expressible by operations; instead, differential equations are often used. These contain derivatives of state variables. Because the derivative of a function at a certain time point t cannot be determined if only the value of this function at t is known, the variables are usually interpreted not as elements of Val , as in the discrete case, but as functions $T \rightarrow Val$ (cf. the topological approach of [8]) or $I \rightarrow Val$ with a non-empty interval $I \subseteq T$ (cf. Hybrid Temporal Logic [6], Hybrid Automata [1], (Extended) Duration Calculus [10]). In most of these approaches, the derivation is a built-in operator that may be applied only to system variables and not to arbitrary expressions because the latter may represent nondifferentiable behavior. The possibilities for enlarging the existing specification means are very limited, especially compared with discrete specification languages. One of the reasons for this restriction is the difficulty of guaranteeing the semantic compatibility of the newly introduced operators with the underlying semantics because the semantic space (often chosen as the set of piecewise differentiable functions) is normally not closed against the user-defined operators (cf. [10, p. 18]).

From the point of view of logics, the derivation is a logical function, and equality is a logical predicate, so a differential equation can be seen as an ordinary (atomic) logical formula. If an appropriate interpretation of the system variables can be found such that the derivation is definable as an ordinary *total* logical function on this interpretation set, then the derivation can be removed from the set of built-in operators of a specification language and replaced by a conventional user interface, allowing the definition of additional logical functions and predicates. In doing so, a substantial advantage with respect to the flexibility of the specification language can be achieved because additional definitions are not only restricted to derivation.

In this paper, the *Continuous Environment-Based Logic (CEL)* is presented. The syntax of CEL is the syntax of the ordinary first-order logic. The main par-

ticularity is the special interpretation of the individual variables as *environment-based values* from the semantic space, called Val_E . The state space of the system is thus equal to $V \rightarrow Val_E$. On the one hand, this interpretation allows definition of the derivation, the limit, the continuity, and other well-known notions of calculus as conventional *total* logical functions and predicates on the elements of Val_E . As the semantic space is closed against all these user-defined constructs and they are all total, each syntactic CEL-term has a well-defined semantics. On the other hand, continuous systems can be specified using CEL, without explicitly mentioning the time variable and without interpreting the variables as functions of time, because the elements of Val_E contain only local information.

The paper is organized as follows. In Section 2, we introduce the logic CEL, describing in particular its syntax, two different interpretations of the syntax, and the relation between these interpretations. Section 3 presents examples of user-defined functions and predicates and illustrates how they can be used to specify continuous systems. Some concluding remarks are given in Section 4.

2 The Logic CEL

2.1 Semantic Space

To motivate the structure of the environment-based values, we consider the following problem. Let $f : \mathbb{R} \rightarrow \mathbb{R}$ be a function and $t \in \mathbb{R}$. Which information about f is necessary and sufficient to decide the following questions: Is f continuous at t ? Does the limit (derivative) of f at t exist and, if so, what is its value? On the one hand, we obviously do not have to know the values of f on whole \mathbb{R} . On the other hand, it is not enough to know only the value of f at t , i.e. $f(t)$. The knowledge of f in every ε -environment of t is sufficient, but for no concrete ε -environment is it really necessary. So, roughly speaking, we can represent the local behavior of f around t by the collection of *all* functions matching pairwise on some ε -environment of t .

To formalize this idea, we define the set $\mathbf{BF} := \mathbb{R} \leftrightarrow Val$ of *basic functions*, which play the role of f in the above motivation. Because in CEL we want to be able to define limit, derivation, continuity, and other similar *environment-based notions* as total functions and predicates on Val_E and because such functions do not always yield a defined value when defined conventionally, we allow basic functions to be partial (denoted by \leftrightarrow). Next, we define the equivalence relation \sim on $\mathbf{BF} \times \mathbb{R}$. $(f_1, t_1) \sim (f_2, t_2)$ states that f_1 and f_2 behave equally in an ε -environment of t_1 and t_2 , respectively. (We use the syntax of Z [11] to express mathematics. \triangleleft denotes the domain restriction of a function. $\{a, b\}$ stands for an open real-valued interval bounded by a and b . $\mathbb{R}_{>0}$ denotes the positive real numbers. The application of the auxiliary function *Shift* on (f, x) shifts to the right the basic function f for the value of x .)

$$\begin{aligned} & - \sim - : (\mathbf{BF} \times \mathbb{R}) \leftrightarrow (\mathbf{BF} \times \mathbb{R}) \\ & \forall f_1, f_2 : \mathbf{BF}; t_1, t_2 : \mathbb{R} \bullet (f_1, t_1) \sim (f_2, t_2) \Leftrightarrow \\ & \quad \exists \varepsilon : \mathbb{R}_{>0} \bullet \{t_1 - \varepsilon, t_1 + \varepsilon\} \triangleleft f_1 = \text{Shift}(\{t_2 - \varepsilon, t_2 + \varepsilon\} \triangleleft f_2, t_1 - t_2) \end{aligned}$$

An *environment-based value* is represented by a function mapping each time point $t \in \mathbb{R}$ to a set of all basic functions matching pairwise on some ε -environment of t ($\text{dom } f$ denotes the domain of f , \mathbb{P}_1 the set of non-empty subsets). We model the local behavior by *all* functions and for *all* points of time in order to avoid different representations for the same local behavior. An environment-based value contains less information than a definition of a function on any nonempty interval and more than a conventional point-based value. We do not require any analytical restrictions on the values of Val ; throughout this paper *continuous systems* are seen as the set of all systems defined on a continuous time domain.

$$\text{Val}_E := \{ ev : \mathbb{R} \rightarrow \mathbb{P}_1 \mathbf{BF} \mid (\forall t_1, t_2 : \mathbb{R} \bullet \forall f_1 : ev(t_1); f_2 : ev(t_2) \bullet \\ (f_1, t_1) \sim (f_2, t_2) \wedge \\ (\forall f : \mathbf{BF} \bullet (f, t_1) \sim (f_1, t_1) \Rightarrow f \in ev(t_1))) \}$$

The auxiliary function $\text{CreateEnv} : \mathbf{BF} \times \mathbb{R} \rightarrow \text{Val}_E$, which is needed for later developments, gets a basic function and a time point as arguments, and yields the environment-based value characterized by this pair. Formally, it is (uniquely) defined by

$$\text{CreateEnv}(f, t) := (\mu ev : \text{Val}_E \mid f \in ev(t)).$$

2.2 Syntax

As mentioned above, the syntax of CEL does not differ from the syntax of ordinary first-order logic. We specify it for the sake of completeness.

Definition 1 (individual variables, signature). We denote the (countable) set of *individual variables* by V . A *signature* S is a triple (F_S, P_S, α_S) , where F_S is a finite or countable infinite set of *function symbols*, P_S a finite or countable infinite set of *predicate symbols* with $F_S \cap P_S = \emptyset$, and $\alpha_S : P_S \cup F_S \rightarrow \mathbb{N}$ the *arity function*. A $p \in P_S$ with $\alpha_S(p) = n$ is called an *n-ary predicate symbol*, and an $f \in F_S$ with $\alpha(f) = n$ an *n-ary function symbol*. If $\alpha(f) = 0$, then we call f a *constant*.

Definition 2 (terms). The set T_S of *S-terms* over a signature S is defined as the smallest set with the following properties:

- All individual variables $v \in V$ are terms.
- All function symbols $c \in F_S$ with arity 0 are terms.
- If $f \in F_S$ is an n -ary function symbol ($n > 0$) and t_1, \dots, t_n are terms, then $f(t_1, \dots, t_n)$ is also a term.

Definition 3 (atomic formulas). The set of *atomic formulas* AtFor_S over a signature S is the smallest set with the following properties:

1. Every 0-ary predicate symbol $p \in P_S$ is an atomic formula.

2. If $p \in P$ ($n > 0$) is an n -ary predicate symbol and t_1, \dots, t_n are terms, then $p(t_1, \dots, t_n)$ is an atomic formula.

Definition 4 (formulas). The set of formulas For_S over a signature S is the smallest set with the following properties:

1. $AtFor_S \subseteq For_S$ (each atomic formula is a formula).
2. Let $A, B \in For_S$. Then $(\neg A)$, $(A \Rightarrow B)$ and $(\forall v \bullet A)$ are elements of For_S .

The parenthesis may be omitted according to the usual priority rules. We use the common logical abbreviations: $A \vee B$ stands for $\neg A \Rightarrow B$, $A \wedge B$ for $\neg(\neg A \vee \neg B)$, and $\exists v \bullet A$ for $\neg \forall v \bullet \neg A$.

2.3 Environment-Based Interpretation

The semantics of CEL is developed according to the same standard structural definitions as the semantics of ordinary first-order logic. The primary difference is the interpretation of individual variables not on arbitrary carrier sets, but on the set Val_E . Since we give another interpretation to the same syntax later (the so-called *function-based interpretation*, cf. Section 2.4), we mark the notions introduced in this section with an E . We denote the sets $Val_E^n \rightarrow Val_E$ and $\mathbb{P}(Val_E^n)$ of n -ary functions and relations upon Val_E for $n \in \mathbb{N}_1$ by \mathcal{F}_n^E and \mathcal{P}_n^E , respectively.

Definition 5 (E-interpretation). Let S be a signature. An *E-Interpretation* (environment-based interpretation) I_E is defined as follows:

1. I_E assigns to each n -ary function symbol f with $n > 0$ an n -ary function on E -values, i.e. $I_E(f) \in \mathcal{F}_n^E$.
2. I_E assigns to each constant $c \in F_S$ an E -value, i.e. $I_E(c) \in Val_E$.
3. I_E assigns to each n -ary predicate symbol p with $n > 0$ an n -ary relation on E -values, i.e. $I_E(p) \in \mathcal{P}_n^E$.
4. I_E assigns to each 0-ary predicate p one of the Boolean values **tt** or **ff**.

Definition 6 (E-evaluation of terms). Let S be a signature and I_E an E -interpretation of S . For each *E-assignment* $\beta : V \rightarrow Val_E$, we define the function $\beta_{I_E} : T_S \rightarrow Val_E$ as follows:

1. $\beta_{I_E}(v) := \beta(v)$ for every $v \in V$
2. $\beta_{I_E}(c) := I_E(c)$ for each $c \in F_S$ with $\alpha(c) = 0$
3. $\beta_{I_E}(f(t_1, \dots, t_n)) := I_E(f)(\beta_{I_E}(t_1), \dots, \beta_{I_E}(t_n))$ for $n > 0$, $f \in F_S$, $\alpha_S(f) = n$, $t_1, \dots, t_n \in T_S$

Definition 7 (E-evaluation of formulas). Let S be a signature, I_E an E -interpretation of S , and β an E -assignment of V . We define $\omega_{I_E, \beta} : For_S \rightarrow \{\mathbf{tt}, \mathbf{ff}\}$ by structural induction over the construction of formulas as follows:

1. $\omega_{I_E, \beta}(p(t_1, \dots, t_n)) := \begin{cases} \mathbf{tt} & \text{if } (\beta_{I_E}(t_1), \dots, \beta_{I_E}(t_n)) \in I(p) \\ \mathbf{ff} & \text{otherwise} \end{cases}$

for $p \in P_S$ with $\alpha_S(p) = n > 0$, $t_1, \dots, t_n \in T_S$

$$\omega_{I_E, \beta}(p) := I_E(p), \text{ for } p \in P_S \text{ with } \alpha_S(p) = 0$$

2. $\omega_{I_E, \beta}(\neg B) := \begin{cases} \mathbf{tt} & \text{if } \omega_{I_E, \beta}(B) = \mathbf{ff} \\ \mathbf{ff} & \text{if } \omega_{I_E, \beta}(B) = \mathbf{tt} \end{cases}$
3. $\omega_{I_E, \beta}(B \Rightarrow C) := \begin{cases} \mathbf{ff} & \text{if } \omega_{I_E, \beta}(B) = \mathbf{tt} \text{ and } \omega_{I_E, \beta}(C) = \mathbf{ff} \\ \mathbf{tt} & \text{otherwise} \end{cases}$
4. $\omega_{I_E, \beta}(\forall v \bullet B) := \begin{cases} \mathbf{tt} & \text{if for all } ev \in Val_E \ \omega_{I_E, \beta^{ev}}(B) = \mathbf{tt} \text{ holds} \\ \mathbf{ff} & \text{otherwise} \end{cases}$

$\beta_v^{ev} : V \rightarrow Val_E$ is defined as follows:

$$\beta_v^{ev}(x) := \begin{cases} ev & \text{if } x = v \\ \beta(x) & \text{if } x \neq v \end{cases}$$

Definition 8 (E-validity of formulas). Let S be a signature and I_E an E-interpretation of S . The partial function $\omega_{I_E} : For_S \rightarrow \{\mathbf{tt}, \mathbf{ff}\}$ is defined as follows:

$$\omega_{I_E}(A) := \begin{cases} \mathbf{tt} & \text{if for all } \beta : V \rightarrow Val_E \ \omega_{I_E, \beta}(A) = \mathbf{tt} \text{ holds} \\ \mathbf{ff} & \text{if for all } \beta : V \rightarrow Val_E \ \omega_{I_E, \beta}(A) = \mathbf{ff} \text{ holds} \\ \text{undefined} & \text{otherwise} \end{cases}$$

2.4 Function-Based Interpretation

The elements of Val_E have a rather complex structure, thus it would be very awkward to define all the functions and predicates the user may need for the specification of continuous behavior on the set Val_E . The *function-based interpretation* presented in this section interprets the individual variables of CEL on basic functions. The function symbols are interpreted not as arbitrary functions $f : \mathbf{BF}^n \rightarrow \mathbf{BF}$, but as *admitted functions* which preserve \sim_n (\sim_n is the extension of \sim to n -dimensional vectors of functions). The predicate symbols are interpreted as *admitted predicates*, defined as subsets of $\mathbf{BF}^n \times \mathbb{R}$ which are closed against \sim_n . All these structures are much more familiar to the user, compared with their E-based counterparts.

The result of an F-evaluation of a formula depends not only on the interpretation I_F and an assignment β , but — unlike E-interpretations — also on the current time point t . A formula is F-valid if it evaluates to true for all interpretations, assignments, *and* time points. Thus, compared with an E-interpretation, an F-interpretation is more intuitive but less abstract because of the explicit dependence on time. In Section 2.5, it will be shown that, under some circumstances, the E-validity and the F-validity are equivalent.

Definition 9 (admitted functions and predicates). The families of *admitted functions* $(\mathcal{F}_n^F \mid n \in \mathbb{N}_1)$ and *admitted predicates* $(\mathcal{P}_n^F \mid n \in \mathbb{N}_1)$ are defined as follows:

- $\mathcal{F}_n^F := \{F : \mathbf{BF}^n \rightarrow \mathbf{BF} \mid$
 $\forall f_1, f_2 : \mathbf{BF}^n; t_1, t_2 : \mathbb{R} \bullet (f_1, t_1) \sim_n (f_2, t_2) \Rightarrow (F(f_1), t_1) \sim (F(f_2), t_2)\}$
- $\mathcal{P}_n^F := \{P : \mathbb{P}(\mathbf{BF}^n \times \mathbb{R}) \mid$
 $\forall f_1, f_2 : \mathbf{BF}^n; t_1, t_2 : \mathbb{R} \bullet$
 $(f_1, t_1) \in P \wedge (f_1, t_1) \sim_n (f_2, t_2) \Rightarrow (f_2, t_2) \in P\}$

It can be shown that the family of admitted functions is closed against composition. More precisely, if $F \in \mathcal{F}_n^F$ and $G \in \mathcal{F}_m^F$, then for each $h_l : \mathbf{BF}^{n+m-1} \rightarrow \mathbf{BF}$ ($l \leq n$) defined by

$$h_l(f_1, \dots, f_{n+m-1}) := F(f_1, \dots, f_{l-1}, G(f_l, \dots, f_{l+m-1}), f_{l+m}, \dots, f_{n+m-1}),$$

$h_l \in \mathcal{F}_{n+m-1}^F$ holds. Moreover, it can be shown that the sets \mathcal{P}_n^F of admitted predicates are closed against union, intersection, and complement.

Definition 10 (F-interpretation). Let $S = (F_S, P_S, \alpha_S)$ be a signature. An *F-interpretation* (function-based interpretation) I_F has the following properties:

1. I_F assigns to each n -ary function symbol f with $n > 0$ an n -ary admitted function, i.e. $I_F(f) \in \mathcal{F}_n^F$.
2. I_F assigns to each constant $c \in F_S$ a *constant* basic function $I_F(c) \in \mathbf{BF}$.
3. I_F assigns to each n -ary predicate symbol p with $n > 0$ an n -ary admitted relation, i.e. $I_F(p) \in \mathcal{P}_n^F$.
4. I_F assigns to each 0-ary predicate symbol p one of the two Boolean values **tt** or **ff**.

Definition 11 (F-evaluation of terms). Let S be a signature and I_F an F-interpretation of S . For every *F-assignment* $\beta : V \rightarrow \mathbf{BF}$ we define the function $\beta_{I_F} : T_S \rightarrow \mathbf{BF}$ as follows:

1. $\beta_{I_F}(v) := \beta(v)$ for all $v \in V$
2. $\beta_{I_F}(c) := I_F(c)$ for all $c \in F_S, \alpha_S(c) = 0$
3. $\beta_{I_F}(f(t_1, \dots, t_n)) := I_F(f)(\beta_{I_F}(t_1), \dots, \beta_{I_F}(t_n))$ for $n > 0, f \in F_S, \alpha_S(f) = n, t_1, \dots, t_n \in T_S$

Definition 12 (F-evaluation of formulas). Let S be a signature, I_F an F-interpretation of S , β an F-assignment of V , and $t \in \mathbb{R}$. We define $\omega_{I_F, \beta, t} : \text{For}_S \rightarrow \{\mathbf{tt}, \mathbf{ff}\}$ by structural definition over the construction of formulas as follows:

1.
$$\omega_{I_F, \beta, t}(p(T_1, \dots, T_n)) := \begin{cases} \mathbf{tt} & \text{if } ((\beta_{I_F}(T_1), \dots, \beta_{I_F}(T_n)), t) \in I(p) \\ \mathbf{ff} & \text{otherwise} \end{cases}$$

for $p \in P_S$ with $\alpha_S(p) = n > 0, T_1, \dots, T_n \in T_S$

$$\omega_{I_F, \beta, t}(p) := I_F(p) \text{ for } p \in P_S \text{ with } \alpha_S(p) = 0$$

2.
$$\omega_{I_F, \beta, t}(\neg B) := \begin{cases} \text{tt} & \text{if } \omega_{I_F, \beta, t}(B) = \text{ff} \\ \text{ff} & \text{if } \omega_{I_F, \beta, t}(B) = \text{tt} \end{cases}$$
3.
$$\omega_{I_F, \beta, t}(B \Rightarrow C) := \begin{cases} \text{ff} & \text{if } \omega_{I_F, \beta, t}(B) = \text{tt} \text{ and } \omega_{I_F, \beta, t}(C) = \text{ff} \\ \text{tt} & \text{otherwise} \end{cases}$$
4.
$$\omega_{I_F, \beta, t}(\forall v \bullet B) := \begin{cases} \text{tt} & \text{if for all } bf \in \mathbf{BF} \ \omega_{I_F, \beta_v^{bf}, t}(B) = \text{tt} \text{ holds} \\ \text{ff} & \text{otherwise} \end{cases}$$

$\beta_v^{bf} : V \rightarrow \mathbf{BF}$ is defined as follows:

$$\beta_v^{bf}(x) := \begin{cases} bf & \text{if } x = v \\ \beta(x) & \text{if } x \neq v \end{cases}$$

Definition 13 (F-validity of formulas). Let S be a signature and I_F an F-interpretation of S . The partial function $\omega_{I_F} : \text{For}_S \rightarrow \{\text{tt}, \text{ff}\}$ is defined as follows:

$$\omega_{I_F}(A) := \begin{cases} \text{tt} & \text{if} & \begin{array}{l} \text{for all } \beta : V \rightarrow \mathbf{BF} \text{ and} \\ \text{for all } t \in \mathbb{R} \ \omega_{I_F, \beta, t}(A) = \text{tt} \text{ holds} \end{array} \\ \text{ff} & \text{if} & \begin{array}{l} \text{for all } \beta : V \rightarrow \mathbf{BF} \text{ and} \\ \text{for all } t \in \mathbb{R} \ \omega_{I_F, \beta, t}(A) = \text{ff} \text{ holds} \end{array} \\ \text{undefined} & \text{otherwise} \end{cases}$$

2.5 Relation Between E-Interpretations and F-Interpretations

As mentioned above, under special circumstances the E-validity and the F-validity coincide. In this section, we define the notion of compatible interpretations and show that under such interpretations this assertion is true. Before doing this we define the *lifting operators*, which allow the definition of logical constructs using basic functions \mathbf{BF} instead of Val_E , thus switching from a complex to a much simpler structure. These \mathbf{BF} -definitions can then be implicitly lifted to Val_E .

The operators LF_n^p and LP_n^p , defined below, lift point-based functions and predicates defined on \mathbb{R} , like $+$, $-$, \cdot , $<$, \leq etc., to (admitted) functions and predicates on basic functions (cf. the upper part of the diagram in Fig. 1).

Definition 14 (lifting of point-based functions and predicates). We define the families of lifting operators for point-based functions ($\text{LF}_n^p \mid n \in \mathbb{N}_1$) and predicates ($\text{LP}_n^p \mid n \in \mathbb{N}_1$) as follows:

- $\text{LF}_n^p : (\text{Val}^n \rightarrow \text{Val}) \rightarrow \mathcal{F}_n^F$
- $\forall F : (\text{Val}^n \rightarrow \text{Val}); f : \mathbf{BF}^n \bullet \text{LF}_n^p(F)(f) = \{(t, F(f_1(t), \dots, f_n(t))) \mid t \in \bigcap_{i=1}^n \text{dom } f_i \wedge (f_1(t), \dots, f_n(t)) \in \text{dom } F\}$
- $\text{LP}_n^p : \mathbb{P}(\text{Val}^n) \rightarrow \mathcal{P}_n^F$
- $\forall P : \mathbb{P}(\text{Val}^n) \bullet \text{LP}_n^p(P) = \{((f_1, \dots, f_n), t) \mid t \in \bigcap_{i=1}^n \text{dom } f_i \wedge (f_1(t), \dots, f_n(t)) \in P\}$

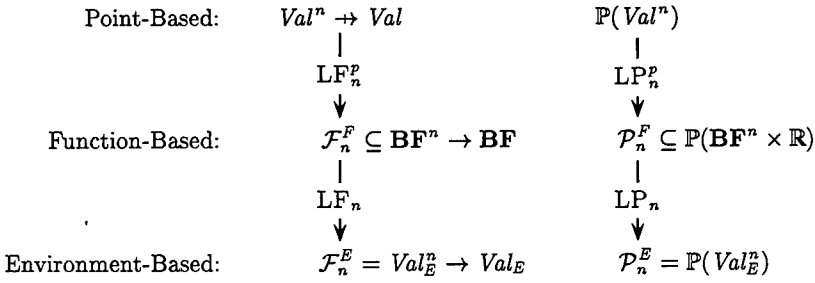


Figure 1. Effect of the lifting operators

The operators LF_n and LP_n lift admitted functions and predicates defined on basic functions to functions and predicates defined on Val_E (cf. the lower part of the diagram in Fig. 1).

Definition 15 (lifting operators). We define the families of lifting operators for functions ($LF_n \mid n \in \mathbb{N}_1$) and predicates ($LP_n \mid n \in \mathbb{N}_1$) by the following axioms (the function $CreateEnv_n$ is the extension of $CreateEnv$ (cf. Sec. 2.1) to the n -dimensional vectors of functions):

- $LF_n : \mathcal{F}_n^F \rightarrow \mathcal{F}_n^E$
 $\forall F : \mathcal{F}_n^F; f : \mathbf{BF}^n; t : \mathbb{R} \bullet LF_n(F)(CreateEnv_n(f, t)) = CreateEnv(F(f), t)$
- $LP_n : \mathcal{P}_n^F \rightarrow \mathcal{P}_n^E$
 $\forall P : \mathcal{P}_n^F; f : \mathbf{BF}^n; t : \mathbb{R} \bullet (f, t) \in P \Leftrightarrow CreateEnv_n(f, t) \in LP_n(p)$

It can be shown that LF_n and LP_n are well-defined [4].

Definition 16 (compatible interpretations). Let S be a signature, I_F an F-interpretation of S , and I_E an E-interpretation of S . We say I_F and I_E are *compatible* if the following holds:

1. For all $c \in F_S$ with $\alpha_S(c) = 0$ $I_F(c) \in I_E(c)(t)$ for all $t \in \mathbb{R}$
2. For all $f \in F_S$ with $\alpha_S(f) = n > 0$ $I_E(f) = LF_n(I_F(f))$
3. For all $p \in P_S$ with $\alpha_S(p) = 0$ $I_E(p) = I_F(p)$
4. For all $p \in P_S$ with $\alpha_S(p) = n > 0$ $I_E(p) = LP_n(I_F(p))$

The definition states that a 0-ary function symbol c is mapped by I_E on an E-value that represents the constant behavior of the function $I_F(c)$. For each F-interpretation there exists (exactly) one compatible E-interpretation. The reverse is not true, because an E-interpretation can assign to 0-ary function symbols E-values, which does not constitute constant behavior. Therefore, there are more E-interpretations than F-interpretations.

The lifting operators LF_n and LP_n allow the definition of logical E-constructs in the straightforward manner. But so far there is no guarantee that the meaning

of E-constructs defined in this way is preserved by the lifting operators. Hence, we cannot execute the usual logical tasks, like formula manipulation or deduction, in E-interpreted CEL with the F-meaning of the constructs in mind. The following theorem shows that, under compatible interpretations, the F-validity and the E-validity are equivalent, thus proving the correctness of using lifted F-constructs in E-interpretations (we need both claims because ω_{I_F} and ω_{I_E} may be undefined (cf. Def. 13 and 8)).

Theorem 17. *Let S be a signature, I_F an F-interpretation of S , I_E an E-interpretation of S that is compatible with I_F , and A an S -formula. Then, the following holds:*

1. $\omega_{I_F}(A) = \mathbf{tt}$ if and only if $\omega_{I_E}(A) = \mathbf{tt}$
2. $\omega_{I_F}(A) = \mathbf{ff}$ if and only if $\omega_{I_E}(A) = \mathbf{ff}$

Proof (sketch)

1. Let β', β'' be F-assignments of V , $T \in T_S$ a term, and $t', t'' \in \mathbb{R}$. Using structural induction, it can be proved that \sim distributes through terms and formulas under F-interpretations:

- (a) $(\forall v : V \bullet (\beta'(v), t') \sim (\beta''(v), t'')) \Rightarrow (\beta'_{I_F}(T), t') \sim (\beta''_{I_F}(T), t'')$
- (b) $(\forall v : V \bullet (\beta'(v), t') \sim (\beta''(v), t'')) \Rightarrow \omega_{I_F, \beta', t'}(A) = \omega_{I_F, \beta'', t''}(A)$

With these results, it can be shown that, if a formula F-evaluates to true for one fixed time point and for all F-assignments, then this formula is F-valid:

$$(\exists t : \mathbb{R} \bullet \forall \beta : V \rightarrow \mathbf{BF} \bullet \omega_{I_F, \beta, t}(A) = \mathbf{tt}) \Leftrightarrow \omega_{I_F}(A) = \mathbf{tt}$$

The analogous result holds for wrong formulas.

2. Let β be an F-assignment of V , $T \in T_S$ a term, and $t \in \mathbb{R}$. $\beta^t : V \rightarrow \text{Val}_E$ denotes the E-assignment corresponding to β and t . It is defined by $\beta^t(v) := \text{CreateEnv}(\beta(v), t)$. The relation between corresponding assignments under compatible interpretations regarding terms and formulas is expressed by the following two facts, which can be proved by structural induction over the construction of terms and formulas, respectively:
 - (a) $\text{CreateEnv}(\beta_{I_F}(T), t) = \beta^t_{I_E}(T)$
 - (b) $\omega_{I_F, \beta, t}(A) = \omega_{I_E, \beta^t}(A)$
3. With the results from 1. and 2., the assertions of the theorem can be proved in a few steps.

3 Specification Examples

In this section, we illustrate how the logic CEL can be used to describe continuous systems. As CEL does not contain any built-in functions and predicates, we must first introduce the required concepts. This is done in Section 3.1. In Section 3.2, these concepts are employed to specify two small continuous systems using the syntax of ZimOO [4], an object-oriented specification language for hybrid systems whose semantics is based on CEL.

3.1 User-Defined Concepts

When specifying continuous systems with CEL, the E-interpretation is assumed because it is much more abstract compared with the F-interpretation. However, it would be very awkward to define all the functions and predicates the user may need directly on the set Val_E . Fortunately, admitted functions \mathcal{F}_n^F and predicates \mathcal{P}_n^F (cf. Def. 9), together with the lifting operators LF_n and LP_n (cf. Def. 15), provide a sound interface to the E-interpretation of CEL. Thus, we are allowed to specify the required logical constructs as elements of \mathcal{F}_n^F or \mathcal{P}_n^F in an intuitively comprehensive manner. When used in the specifications of continuous systems, these functions and predicates are implicitly lifted to elements of \mathcal{F}_n^E or \mathcal{P}_n^E , respectively. Theorem 17 ensures that the intuitive meaning is not violated. Here, we use the syntax of Z instead of conventional mathematics to introduce logical constructs.

Functions and Predicates The construct introduced first is the unary predicate δ_P which characterizes points with defined local behavior ($=$ is the definition symbol). *const* describes constant local behavior. It is obvious that δ_P and *const* are admitted predicates. Thus, their liftings can be used in continuous specifications.

$$\begin{aligned}\delta_{P_-} &== \{f : \mathbf{BF}; t : \mathbb{R} \mid t \in \text{dom } f\} \\ \text{const}_- &== \{f : \mathbf{BF}; t : \mathbb{R} \mid (\exists \varepsilon : \mathbb{R}_{>0}; v : \text{Val} \bullet \{t - \varepsilon, t + \varepsilon\} \subseteq \text{dom } f \wedge \\ &\quad \text{ran}(\{t - \varepsilon, t + \varepsilon\} \triangleleft f) = \{v\})\}\end{aligned}$$

The next three functions define the limit from the left, the limit from the right, and the “ordinary” limit. As the set *Val* must meet certain requirements to allow the definition of limit (it should be at least a metric space), we interpret *Val* henceforth as the set of real numbers. In the definition, we use the type seq_∞ and the function lim_{seq} , which are not defined here. They denote the type of infinite sequences and the limit of sequences, respectively. We consider a function to be a set of pairs — a view familiar to Z users. It can easily be proved that all the three limits are admitted functions.

$$\begin{array}{|l} \rightarrow, \leftarrow, \overleftrightarrow{} : \mathbf{BF} \rightarrow \mathbf{BF} \\ \hline \forall f : \mathbf{BF} \bullet \\ \quad \vec{f} = \{x, l : \mathbb{R} \mid (\text{let } SEC == \text{seq}_\infty \{t : \text{dom } f \mid t < x\} \bullet \\ \quad (\exists s : SEC \bullet \text{lim}_{\text{seq}} s = x) \wedge \\ \quad (\forall s : SEC \bullet \text{lim}_{\text{seq}} s = x \Rightarrow \text{lim}_{\text{seq}}(\lambda n : \mathbb{N}_1 \bullet f(s \ n)) = l))\} \\ \quad \overleftarrow{f} = \{x, l : \mathbb{R} \mid (\text{let } SEC == \text{seq}_\infty \{t : \text{dom } f \mid t > x\} \bullet \\ \quad (\exists s : SEC \bullet \text{lim}_{\text{seq}} s = x) \wedge \\ \quad (\forall s : SEC \bullet \text{lim}_{\text{seq}} s = x \Rightarrow \text{lim}_{\text{seq}}(\lambda n : \mathbb{N}_1 \bullet f(s \ n)) = l))\} \\ \quad \overleftrightarrow{f} = \vec{f} \cap \overleftarrow{f} \end{array}$$

that (f, t) is not required to fulfill pr . *EnvPoint* maps admitted predicates to admitted ones.

$$\begin{array}{|l}
 \text{EnvPoint} : \mathbb{P}(\mathbf{BF} \times \mathbb{R}) \rightarrow \mathbb{P}(\mathbf{BF} \times \mathbb{R}) \\
 \hline
 \forall pr : \mathbb{P}(\mathbf{BF} \times \mathbb{R}) \bullet \\
 \text{EnvPoint } pr = \{f : \mathbf{BF}; t : \mathbb{R} \mid \\
 \neg \exists T : \text{seq}_{\infty}(\mathbb{R} \setminus \{t\}) \bullet \lim_{\text{seq}} T = t \wedge \\
 (\forall i : \text{dom } T \bullet (f, T \ i) \notin pr)\}
 \end{array}$$

The most general data type *BASIC* contains all total local behaviors.

$$BASIC == \delta_{P-} \cap \text{EnvPoint}(\delta_{P-})$$

The following definition introduces the data types used in the specifications in Section 3.2. *LIM* denotes the total local behaviors with existing limits from the left and the right. The frequently used data type *SEM* characterizes piecewise differentiable local behavior, i.e. local behavior without accumulation of nondifferentiable points. *CONT* and *DIFF* denote continuous and differentiable behavior, respectively. The data type *CONST* describes the constant behavior. *STEP* models step functions defined on a continuous time domain. Finally, *CLOCK* describes differentiable local behavior with the gradient 1.

$$\begin{aligned}
 LIM &== BASIC \cap \vec{\mathcal{L}}_- \cap \vec{\mathcal{L}}_- \\
 SEM &== LIM \cap \text{EnvPoint}(\mathcal{D}_-) \\
 CONT &== SEM \cap \mathcal{C}_- \\
 DIFF &== SEM \cap \mathcal{D}_- \\
 CONST &== BASIC \cap \text{const}_- \\
 STEP &== BASIC \cap \text{EnvPoint}(\text{const}_-) \cap (\vec{\mathcal{L}}_- \cup \vec{\mathcal{L}}_-) \\
 CLOCK &== \{f : \mathbf{BF}; t : \mathbb{R} \mid (f, t) \in DIFF \wedge \dot{f}(t) = 1\}
 \end{aligned}$$

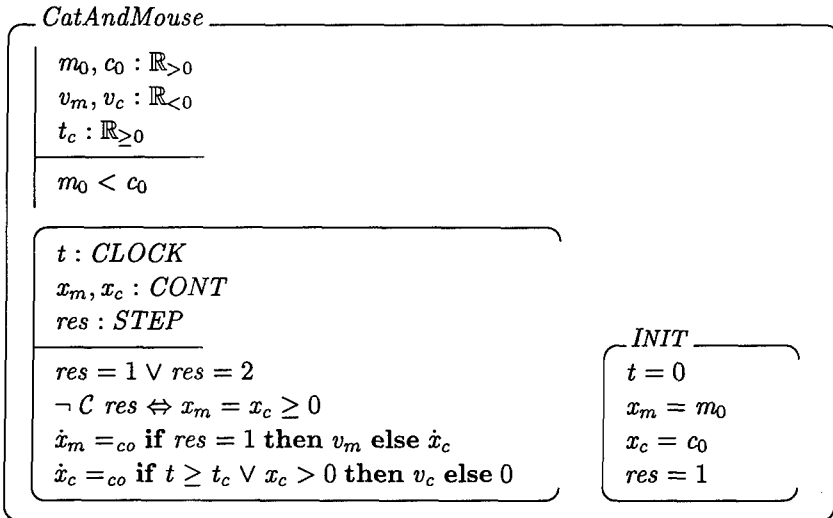
3.2 Examples of Continuous Systems

As mentioned above, the logic CEL was used to describe the semantics of the continuous classes of ZimOO [4], an object-oriented specification language for hybrid systems. ZimOO is based on Object-Z [3], an object-oriented extension of Z [11]. It extends Object-Z, allowing descriptions of the discrete and continuous features of a system in a common formalism. ZimOO supports three different kinds of classes: discrete, continuous, and hybrid. We use the syntax of the continuous ZimOO classes to give some examples of CEL-specifications.

Axioms are used to specify the state space of continuous ZimOO classes. They are formulated using the syntax of first-order logic and interpreted as CEL-formulas, the E-interpretation being assumed. There are no built-in logical functions or predicates in the kernel of ZimOO. Instead, we use the functions and predicates defined in the previous subsection (as justified at the beginning

of Section 3.1, the functions and predicates defined there may be used in E-interpreted CEL-formulas and therefore in ZimOO classes). Additionally, we use the common point-based functions and predicates defined on reals like $+$, $-$, \leq , etc. They can all be lifted to \mathcal{F}_n^E or \mathcal{P}_n^E by the composition of LF_n^P and LF_n or by the composition of LP_n^P and LP_n , respectively. In particular, the equality on reals is lifted to \mathcal{F}_2^E . Note that, consequently, we use a point-based equality which depends only on the current real value of the expressions involved, neglecting their local environments.

Cat and Mouse The cat-and-mouse-problem [9] is a simple benchmark from the area of real-time and hybrid systems. We specify it here to demonstrate the description possibilities of languages based on CEL (cf. the class *CatAndMouse*). The example deals with a cat trying to catch a mouse, which in turn attempts to escape into a hole. The problem is one-dimensional, i.e. the cat, the mouse, and the hole are on a straight line, the cat and mouse moving along this line. Initially, the mouse, which is located between the cat and the hole at distance m_0 from the hole, starts running towards the hole at a constant velocity v_m . t_c time units later, the cat, which is positioned at c_0 , starts chasing the mouse at the constant velocity v_c . All these constants are declared as real numbers in the *axiomatic schema* of *CatAndMouse*.



The state space and the dynamics of the system are described in the *state schema* of *CatAndMouse*. As the example contains an explicit delay, we need a clock variable t . The current positions of the mouse and the cat are denoted by the variables x_m and x_c , respectively. The result of the “race” is encoded in the variable res . $res = 1$ means the mouse wins, $res = 2$ means the cat is the winner. When the constants t_c , m_0 , 1, 2 etc. are used in the state schema, their values are implicitly lifted to Val_E , i.e. to CONST . The second axiom states

that the value of res , which initially equals 1, can change if and only if the cat overtakes the mouse before it disappears into the hole. The last two axioms can be interpreted as differential equations describing the movement of the mouse and the cat. Depending on the value of res , x_m behaves according to $\dot{x}_m = v_m$ or $\dot{x}_m = \dot{x}_c =_{co}$ ensures that if res does not jump and \dot{x}_c represents defined local behavior, then the derivative of x_m exists and fulfills one of the two differential equations. In jump points of res and in points where x_c is not differentiable, the value of x_m is uniquely determined by the continuity of x_m .

Billiards As a further example, we specify the billiards game from [2]. The billiard table is assumed to have the length L and the width W . Friction is neglected, i.e. we assume the absolute values of the ball velocities v_x and v_y in the x - and y -directions to be constant. The current position of the ball is described by the pair (x, y) , the velocity directions by d_x and d_y ; the current velocity is therefore given by $(d_x \cdot v_x, d_y \cdot v_y)$. The first implication in the state schema states that the x -velocity v_x may only change its direction d_x if a collision with one of the x -borders occurs. The third implication ensures that such a change takes place when an x -border is reached. The second and the fourth implications describe the same facts for the y -direction.

Note the use of the limit operators in the last two implications. They can be applied not only to individual variables but also to expressions because the (lifted) multiplication operator “ \cdot ” is total on $Val_E \times Val_E$, thus yielding a proper element of Val_E which can be further processed by the limit operators.

Billiards

$L, W, v_x, v_y : \mathbb{R}_{>0}$

$x, y : CONT$

$d_x, d_y : STEP$

$(d_x = -1 \vee d_x = 1) \wedge (d_y = -1 \vee d_y = 1)$

$\dot{x} =_{co} d_x \cdot v_x$

$\dot{y} =_{co} d_y \cdot v_y$

$\neg C \ d_x \Rightarrow x = 0 \vee x = L$

$\neg C \ d_y \Rightarrow y = 0 \vee y = W$

$x = 0 \vee x = L \Rightarrow \overleftarrow{d_x \cdot v_x} = -\overrightarrow{d_x \cdot v_x}$

$y = 0 \vee y = W \Rightarrow \overleftarrow{d_y \cdot v_y} = -\overrightarrow{d_y \cdot v_y}$

INIT

$0 < x < L$

$0 < y < W$

$d_x = d_y = 1$

4 Conclusion

The paper describes the Continuous Environment-Based Logic (CEL) and proposes its use for the specification of continuous components of hybrid systems.

The syntax of CEL is the syntax of first-order logic. The semantic particularity is the interpretation of variables on the set Val_E of environment-based values. The elements of Val_E contain less information than any nonempty interval of a behavioral function, but more than a conventional point-based value, lying more or less in between. Because of this choice of the semantic space, the derivation and other environment-based constructs can be defined directly on E-values, and the interpretation of the variables as functions can be avoided. The definition of additional logical constructs can be performed in a comprehensively intuitive manner with explicit access to the time variable. The lifting operators translate these constructs implicitly into the semantics of CEL. It has been shown that for compatible interpretations the meaning of the constructs is preserved. CEL has been used for the semantics definitions of an object-oriented specification language for hybrid systems. In general, it can be very useful for formal semantics definition of object-oriented simulation languages for dynamic systems.

Acknowledgments

I wish to thank Matthias Weber for his valuable comments. Thanks also to Phil Bacon for polishing up my English.

References

1. R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In Grossman et al. [5], pages 209–229.
2. R. Alur, C. Courcoubetis, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.
3. R. Duke, P. King, G. A. Rose, and G. Smith. The Object-Z specification language: Version 1. Technical Report 91-1, Department of Computer Science, University of Queensland, St. Lucia 4072, Australia, April 1991.
4. V. Friesen. *Objektorientierte Spezifikation hybrider Systeme*. PhD thesis, Technical University of Berlin, 1997.
5. R. Grossman, A. Nerode, H. Rischel, and A. Ravn, editors. *Hybrid Systems*, volume 736 of *LNCIS*. Springer-Verlag, 1993.
6. T. A. Henzinger, Z. Manna, and A. Pnueli. Towards refining temporal specifications into hybrid systems. In Grossman et al. [5], pages 60–76.
7. C. B. Jones. *Systematic Software Development using VDM*. Prentice Hall, second edition, 1990.
8. B. P. Mahony. *The Specification and Refinement of Timed Processes*. PhD thesis, Department of Computer Science, University of Queensland, Australia, 1992.
9. O. Maler, Z. Manna, and A. Pnueli. From timed to hybrid systems. In J. W. de Bakker, K. Huizing, W.-P. de Roever, and G. Rozenberg, editors, *Real Time: Theory in Practice*, volume 600 of *LNCIS*, pages 446–484. Springer-Verlag, 1992.
10. A. P. Ravn. *Design of Embedded Real-Time Computing Systems*. PhD thesis, Technical University of Denmark, Lyngby, 1995.
11. J. M. Spivey. *The Z Notation: A Reference Manual*. Prentice Hall, second edition, 1992.

Integrating Projections

Mark R. Greenstreet
Department of Computer Science

University of British Columbia
Vancouver, BC V6T 1Z4
Canada
mrg@cs.ubc.ca

Ian Mitchell
Scientific Computing and
Computational Mathematics
Stanford University
Stanford, CA 94305-9025
USA
mitchell@scm.stanford.edu

Abstract This paper describes three techniques for reachability analysis for systems modeled by ordinary differential equations (ODEs). First, linear models with regions modeled by convex polyhedra are considered, and an exact algorithm is presented. Next, non-convex polyhedra are considered, and techniques are presented for representing a polyhedron by its projection onto two-dimensional subspaces. This approach yields a compact representation, and allows efficient algorithms from computational geometry to be employed. Within this context, an approximation technique for reducing non-linear ODE models to linear nonhomogeneous models is presented. This reduction provides a sound basis for applying methods for linear systems analysis to non-linear systems.

1 Introduction

We are interested in verifying that circuits, as modeled by systems of non-linear ordinary differential equations (ODE's), correctly implement discrete specifications. Challenging verification problems arise when VLSI designers use methods such as precharged logic, single-phase clocking, and sense-amp based techniques that depend on the analog properties of the circuits to obtain better performance. In current practice, design validation relies heavily on simulation tools such as SPICE [Nag75]. However, even the best model is only approximate, and each simulation run can only consider a particular set of functions as inputs to the circuit and a particular set of values for model parameters. To obtain a reasonable level of confidence in a design, a large number simulations must be run. This process can be extremely time consuming; yet, in the end, simulation can not prove the correctness of a design.

Recently, we have been exploring an alternative approach to the problem of circuit-level design verification, based on ideas from dynamical systems theory. Correctness criteria for a circuit can be formulated in a logic which has meaning in both continuous and discrete domains. Rather than considering individual simulation runs, correctness criteria become topological properties in the continuous domain that must hold for an invariant set that contains *all* possible trajectories of the ODE model. To establish these invariants, we construct regions such that all trajectories on the boundaries flow inward [GM97]. For simple models these regions can be constructed manually, but for models

arising in real circuits more automated methods are required. In [Gre96], we showed how these invariant sets can be constructed by reachability analysis using numerical integration.

An important advantage of our approach is that our analysis is based on ODE models similar to those that are used for industrial circuit simulation. Thus, our results are comparable with those obtained by traditional simulations—by speaking the same language as circuit designers, we encourage interaction with the eventual users of our techniques. Furthermore, considerable effort continues to be invested in developing accurate models for current fabrication processes. By using ODE models as the basis of our work, we can exploit these advances directly.

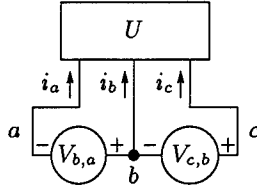
Two contributions are made by this paper. First, we describe an efficient way of representing non-convex high dimensional polyhedra using two-dimensional projections. This representation is by no means universal; however, it has shown promising results for a small number of circuits that we have analysed. Second, we show an integration based approach for computing reachability between regions represented using projections. Although we use floating point arithmetic in our implementation to obtain acceptable performance; in principle, the same techniques could be implemented with rational arithmetic and conservative rounding to create a strictly conservative implementation of the algorithm. The theoretic aspects of these contributions are contained in section 3. Preceding that section is a description of our models.

2 Models

In this section we show how to construct ODE models for our analysis. Section 2.1 describes the construction of models for MOS circuits. These circuits require inputs, and we typically wish to verify a circuit for all legal inputs. Readers familiar with circuit modeling may wish to skip directly to section 2.2, which describes Brockett's annulus construction and shows how it can be used to model inputs to our circuits.

2.1 Circuit Models

We model MOS circuits as a collection of voltage controlled current sources and (linear) capacitors. A voltage controlled current source defines a relationship between the voltages on its terminals and the currents flowing into those terminals. By convention, current is the flow of "positive charges," and a flow of electrons into the device is represented by a negative current. Consider the device depicted below:



The device U is connected to three nodes, a , b , and c . The voltage V_a denotes the voltage at node a , and likewise for V_b and V_c . We write V_{ba} to denote $V_b - V_a$ and likewise for V_{cb} . The current i_a denotes the current flowing into device U through node a . If U is a voltage controlled current source, then i_a is a function of the voltages V_a , V_b and V_c . We write $i_a = U_a(V_a, V_b, V_c)$.

More generally, let \bar{V} denote the vector of node voltages in the circuit, and let \bar{i}_U denote the vector of currents flowing into U through each node of the circuit. We write

$$\bar{i}_U = I_U(\bar{V}) \quad (1)$$

For example, an n-channel MOSFET can be modeled as a three terminal, voltage controlled current source. The three terminals are the gate, g , the source, s , and the drain, d . A simple model (see [GD85], equations 2.85 - 2.87) is

$$\begin{aligned} i_g(V_g, V_s, V_d) &= 0 \\ i_d(V_g, V_s, V_d) &= 0, & \text{if } V_{ds} \geq 0 \text{ \& } V_{gs} < V_t \\ &= G(V_{gs} - V_t)^2, & \text{if } V_{ds} \geq 0 \text{ \& } V_{ds} > V_{gs} - V_t \geq 0 \\ &= GV_{ds}(2(V_{gs} - V_t) - V_{ds}), & \text{if } V_{ds} \geq 0 \text{ and } V_{gs} - V_t \geq V_{ds} \geq 0 \\ &= -i_d(V_g, V_d, V_s), & \text{if } V_{ds} < 0 \end{aligned} \quad (2)$$

$$i_s(V_g, V_s, V_d) = -i_d$$

where V_t is the "threshold voltage" of the transistor, and G is the transconductance. These two constants are determined by the size and shape of the transistor and by properties of the fabrication process.

A capacitor defines a relationship between the time derivatives of the voltages on the terminals and the currents flowing into these terminals. For a capacitor of fixed capacitance C connected to nodes a and b ,

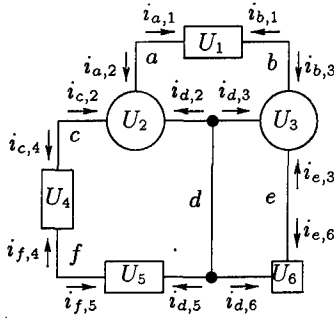
$$i_b = -i_a = C \frac{dV_b}{dt} - C \frac{dV_a}{dt}$$

More generally, a capacitor U defines a matrix valued function C_U such that

$$\bar{i}_U = C_U(\bar{V}) \frac{d\bar{V}}{dt} \quad (3)$$

For the models arising from MOS circuits, this matrix corresponds to a network of voltage dependent, two-terminal capacitors. Physically, there must be some capacitance between every pair of nodes; in practice, many of these capacitances are small and neglected when constructing a circuit model. Any realistic model will associate at least one capacitor with each node; for such models, $C_U(\bar{V})$ is real-symmetric and positive definite.

Given models for each device in the circuit, we construct an ODE model for the whole system using Kirchoff's current law. As depicted in figure 1, Kirchoff's current law states that the sum of the currents flowing into each node of the circuit must be zero. Likewise, the sum of the currents flowing into each device must be zero. Both of these constraints are direct consequences of charge conservation.



Kirchoff's Current Law:

$$\forall x \in \{a \dots f\}, \sum_{m=1}^6 i_{x,m} = 0$$

$$\forall m \in \{1 \dots 6\}, \sum_{x=a}^f i_{x,m} = 0$$

Fig. 1. Kirchoff's Laws

From Kirchoff's current law, we have

$$\sum_{U \in C} C_U(\bar{V}) \frac{d\bar{V}}{dt} + \sum_{U \in I} I_U(\bar{V}) = 0$$

where C denotes the set of capacitor devices, and I denotes the set of current source devices. Solving for $d\bar{V}/dt$ yields

$$\frac{d\bar{V}}{dt} = -(\sum_{U \in C} C_U(\bar{V}))^{-1} (\sum_{U \in I} I_U(\bar{V})) \quad (4)$$

which is an ODE model for the circuit.

The device models above are simplistic, allowing a shorter presentation and making the analysis in the remainder of this paper tractable. While these models capture many of the key features of MOS circuit operation, we note that the transistor model of equation 2 neglects the body effect and short channel effects. Similarly, when modeling capacitors we make the simplifying assumption that C_U is a constant; in real MOS designs, C_U depends substantially on \bar{V} . Kirchoff's current law is itself an approximation of Maxwell's equations, and so ignores "displacement currents." Typically, designers use more accurate models than those presented for transistors and capacitors—here we have chosen to avoid complexity while retaining the key features of realistic circuit models.

2.2 Input Signals

The problem of verifying an entire chip at the ODE level appears to be hopelessly intractable. Instead, we focus on the problem of verifying small circuits

and showing that the outputs of one circuit satisfy the constraints that we assume for inputs to other circuits. Such a method requires a mechanism for specifying the expected inputs and the allowed outputs of each small circuit.

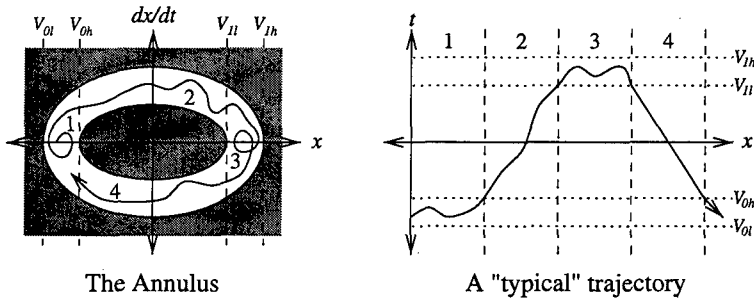


Fig. 2. Brockett's Annulus

Figure 2 depicts the annulus proposed by Brockett [Bro89] that we use to specify the levels and transitions of signals. When a variable is in region 1, its value is constrained but its derivative may be either positive or negative. We will consider this a logically low signal. When the variable leaves region 1, it must enter region 2. Because the derivative of the variable is strictly positive in this region, it makes a monotonic transition rising to region 3. Regions 3 and 4 are analogous to regions 1 and 2, and correspond to logically high and monotonically falling signals respectively. Because transitions through regions 2 and 4 are monotonic, traversals of these regions are distinct events. The properties of the annulus provide a topological basis for discrete behaviours.

Many common signal parameters are represented by the geometry of an annulus. The horizontal radii of the annulus define the maximum and minimum high and low levels of the signal (i.e. V_{0l} , V_{0h} , V_{1l} , and V_{1h} in figure 2). The maximum and minimum rise time for the signal correspond to trajectories along the upper-inner and upper-outer boundaries of the annulus respectively. Likewise, the lower-inner and lower-outer boundaries of the annulus specify the maximum and minimum fall times.

3 Reachability Analysis

In this section we present our theoretic results. After looking at the connection between verification and reachability, we examine three increasingly difficult reachability analyses: linear models with convex polyhedra, linear models with non-convex polyhedra, and finally nonlinear models with non-convex polyhedra.

3.1 Verification as Reachability

Many circuit verification problems can be formulated as reachability analysis problems. For example, consider a circuit that implements a simple state machine. An ODE model provides a mapping between the continuous circuit state (node voltages) and the time derivative of that state. Thus, given a point in the continuous space, the value and derivative of each signal is known. Using a Brockett annulus, each signal can be interpreted discretely as being low, rising, high, or falling. The continuous model implements the discrete specification if every reachable point in the continuous model corresponds to a state or transition of the discrete specification.

First consider the verification of bounded prefixes of trajectories. For a circuit with d nodes, the continuous state space is \mathbb{R}^d . We assume that the derivative function for the model is autonomous (i.e. independent of time) and finitely piecewise continuous (therefore locally bounded). Given a bounded region $Q \subset \mathbb{R}^d$, $Q_0 \subseteq Q$, and $t_f \in \mathbb{R}^+$, we want to show that all trajectories that start in Q_0 at time 0 will remain in Q for all times up to t_f . Our approach to this problem is to construct a sequence of time steps $t_0 < t_1 < \dots < t_k$ such that $t_0 = 0$ and $t_k = t_f$. For $i = 1 \dots k$, we construct a region Q_i such that any trajectory that starts in Q_{i-1} at time t_{i-1} will be in Q_i at time t_i . We then construct a second set of regions Q'_0, \dots, Q'_{k-1} such that any trajectory that starts in Q_i at time t_i will remain in Q'_i up to and including time t_{i+1} . If $\cup_{i=0}^{k-1} Q'_i \subseteq Q$, then all trajectories that start in Q_0 at time 0 will remain in Q for all times up to t_f as can be readily shown by the construction of Q'_i .

Now consider infinite trajectories. Let Q, Q_i, Q'_i , and t_i be constructed as above, $D = \cup_{i=0}^{k-1} Q'_i$, and $Q^+ = \cup_{i=0}^{k-1} Q_i$. If $Q_k \subseteq Q^+$, then any trajectory that starts in Q_0 remains in D forever. To see this, let $x : \mathbb{R}^+ \rightarrow \mathbb{R}^d$ be a trajectory with $x(0) \in Q_0$. Let $\tau_{\min} = \min_{i=1}^k t_k - t_{k-1}$. There exists a sequence of times, τ_m , such that for all $m \geq 0$, $x(\tau_m) \in Q^+$ and $\tau_m \geq m\tau_{\min}$. The proof is completed by induction on m . For $m = 0$, $\tau_m = 0$. For $m > 0$, let $j \in \{0 \dots k-1\}$ such that $x(\tau_{m-1}) \in Q_j$. Let

$$\tau_m = \tau_{m-1} + (t_{j+1} - t_j) \geq \tau_{m-1} + \tau_{\min} \geq m * \tau_{\min}$$

Then, $x(\tau_m) \in Q_{j+1} \subseteq Q^+$.

In general, it is not feasible to represent exactly the reachable regions of systems modeled by ODEs. Most non-linear ODEs, including those that arise when modeling VLSI circuits, do not have closed form solutions. Because proof of safety properties is our objective, over estimation of the reachable space is conservative—false negatives are possible, but not false positives. Consequently, we use “containing approximations”, within which lie the true reachable state spaces.

As described above, the next few sections examine three different cases of reachability analysis. First, we consider the special case of linear ODE's where the initial region is a convex polyhedron—we show that the Q_i sequence can

be computed exactly, and the Q'_i sequence can be computed with arbitrary accuracy. In general, convexity is not preserved by non-linear models, and we develop our treatment of non-linear models in two steps. First, section 3.3 presents a conservative approximation technique for the particular class of non-convex polyhedra that can be represented by their projections onto two-dimensional subspaces; however, linear models are retained. In section 3.4 we show how these projection polyhedra can be used with non-linear models.

3.2 Linear models and convex polyhedra

This section presents the special case where the ODE model is linear, and Q_0 is convex. An ODE model is linear if it can be written in the form

$$\dot{x} = Ax \quad (5)$$

where $x : \mathbb{R}^+ \rightarrow \mathbb{R}^d$ is a trajectory and $A \in \mathbb{R}^{d \times d}$ is a matrix (note that this definition of “linear” is more general than the one used in much of the hybrid systems literature). We assume that A has a full-rank set of eigenvectors. If not, a small perturbation of A will produce such a matrix, and the techniques presented in section 3.4 can be applied. With this assumption, the solution of equation 5 is [HS74]

$$x(t) = e^{tA}x(0) \quad (6)$$

For any fixed value of t , e^{tA} is a linear operator that can be represented by a matrix, and e^{tA} is invertible.

A d -dimensional convex polyhedron with m faces can be represented by linear program of the form

$$Mx \leq B \quad (7)$$

where $M \in \mathbb{R}^{m \times d}$ is a matrix and $B \in \mathbb{R}^m$ is a vector (see [PS82]). We write (M, B) to denote the linear program of equation 7, and write $x \in (M, B)$ to denote that x satisfies this linear program.

Polyhedra can be bloated. If (M, B) is a linear program, and u is a real number, then, $\text{bloat}((M, B), u)$ is the polyhedron obtained by moving each face of M outward by u . Let $\Delta \in \mathbb{R}^d$ be a vector such that its j^{th} element is given by $\Delta_j(j) = u\|M_j\|_2$, where $\|M_j\|_2$ denotes the L2 norm of row j of M . Then,

$$\text{bloat}((M, B), u) = (M, B - \Delta) \quad (8)$$

Convexity is preserved by linear operators. In particular, let the linear program (M_0, B_0) describe the convex region Q_0 , let $t_1 \in \mathbb{R}^+$, and let A be the matrix representation of a linear ODE model. A point x is reachable from Q_0 at time t_1 if and only if $x \in (M_0 e^{-t_1 A}, B)$, which follows directly from

equations 6 and 7. Thus, we can construct $Q_1 \dots Q_k$ such that for $i = 1 \dots k$, any trajectory that starts in Q_{i-1} at time t_{i-1} will be in Q_i at time t_i . In particular,

$$Q_i = (M_0 e^{-t_i A}, B) \quad (9)$$

These Q_i are exact.

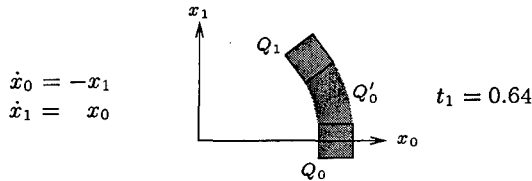


Fig. 3. A simple linear system

Although the Q_i 's (the reachable regions at each time step) are convex, the same does not necessarily hold for the Q'_i 's (the regions reachable during all times between steps). For example, consider the system depicted in figure 3. Trajectories are counter-clockwise circles centered at the origin. Although Q_0 and Q_1 are both convex, the minimal region for Q'_0 is the region swept out by moving Q_0 through an arc of t_1 radians (the shaded region in figure 3). Region Q'_0 is not convex.

Rather than trying to solve for Q'_0 exactly, we will find an approximation. Note that $\dot{x} = Ax$ is locally bounded; therefore it is bounded in Q . Define the scalar $\|\dot{x}\|_{\max} = \max_{x \in Q} \|Ax\|_2$. A trajectory that starts in region Q_i at time t_i remains within a distance $(t_{i+1} - t_i)\|\dot{x}\|_{\max}$ of Q_i until time t_{i+1} . Let

$$Q'_i = \text{bloat}((M_0 e^{-t_i A}, B), \|\dot{x}\|_{\max}) \quad (10)$$

For any trajectory x such that $x(t_i) \in Q_i$ and for any time $t \in [t_i, t_{i+1}]$, $x(t) \in Q'_i$ as required.

Although the Q'_i are containing approximations, each one is computed from an exact Q_i —the errors of making a conservative approximation do not accumulate between time steps. To achieve accurate estimates of the reachable space, the time steps should be relatively small so that there is little of Q'_i outside of $Q_i \cup Q_{i+1}$. For example, this approach would compute a large overestimate of Q'_0 for the time step depicted in figure 3.

A straightforward approach to verification is to construct a sequence of Q_i and Q'_i as described above, and verify that each Q'_i is contained in Q . If all containments are established, then the verification is complete. Otherwise, choose i such that Q'_i is not contained in Q . A counterexample to the verification is established if either of the exact solutions Q_i or Q_{i+1} is not contained in Q . If neither of the exact solutions provide a counterexample,

divide the step from Q_i to Q_{i+1} into two smaller steps and repeat the verification. This process terminates when containment of all the Q'_i 's is verified, a counter-example is found, or the time step is smaller than is meaningful for the chosen model. In the latter case, the property cannot be verified with the given model. Typical variation in MOS circuit parameters can $\pm 20\%$ or more, although closely matched circuits (e.g. sense-amplifiers, see [Bak90]) can be designed that are balanced to within a few parts per thousand.

3.3 Linear models and non-convex polyhedra

Although systems with linear ODE models can be analysed quite accurately using the techniques described in the previous section, such systems do not have a rich enough phase space structure for interesting digital computation. In a linear system, the asymptotic behaviour of trajectories is either convergence towards the origin, divergence to infinity, or an orbit centered at the origin. In order to examine more interesting systems, we need techniques to analyse non-linear models. In general, these models do not preserve the convexity of polyhedra; therefore, we begin by describing the class of non-convex polyhedra that we use in our analysis.

Representation

We represent high dimensional polyhedra by their projections onto two dimensional subspaces, where these projections are not required to be convex. Conversely, a full dimensional polyhedron can be obtained from its projections by back-projecting each into a prism in \mathbb{R}^d and computing the intersection of those prisms (see figure 4). More formally, let $\{u_1, u_2, \dots, u_d\}$ be an orthogonal basis for \mathbb{R}^d . If P is a polygon, we write $(u_{X(P)}, u_{Y(P)})$ to denote the basis of P . We write $\text{ConvexHull}(P)$ to denote the convex hull (see [PS85]) of P , and it is understood that $X(\text{ConvexHull}(P)) = X(P)$ and $Y(\text{ConvexHull}(P)) = Y(P)$. We write $\text{prism}(P)$ to denote the inverse projection of P back into the full dimensional space:

$$\text{prism}(P) = \{(x_1, \dots, x_d) \in \mathbb{R}^d \mid (x_{X(P)}, x_{Y(P)}) \in P\} \quad (11)$$

Let \mathcal{P} be a collection of polygons. The object represented by \mathcal{P} is $Q(\mathcal{P})$ where

$$Q(\mathcal{P}) = \bigcap_{P \in \mathcal{P}} \text{prism}(P) \quad (12)$$

We note that faces of $Q(\mathcal{P})$ correspond to edges of the projection polygons. If P is a projection polygon, and e is an edge of P , we write $X(e)$ and $Y(e)$ to denote $X(P)$ and $Y(P)$ respectively. Likewise, we define $\text{prism}(e)$ to be

$$\text{prism}(e) = \{(x_1, \dots, x_d) \in \mathbb{R}^d \mid (x_{X(e)}, x_{Y(e)}) \in e\} \quad (13)$$

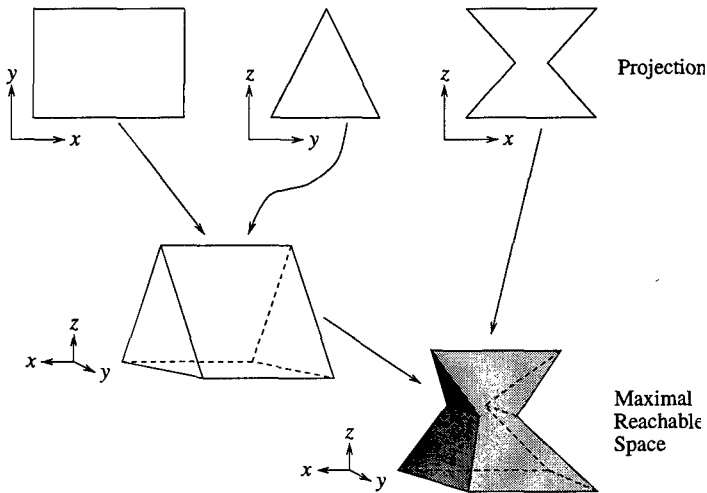


Fig. 4. A three dimensional polyhedron and its projections

If e is an edge of a projection polygon, we write $face(e, \mathcal{P})$ to denote the corresponding edge of e :

$$face(e, \mathcal{P}) = Q(\mathcal{P}) \cap prism(e) \quad (14)$$

We write $face(e)$ when \mathcal{P} is apparent from context.

There are several advantages to this representation. First, it corresponds to an engineer's intuitive notion of how a circuit works. Typically, each signal is "controlled" by a small number of other signals. Pairing each node with each of its controlling nodes naturally captures the causal behaviour of the circuit. Because most circuits have limited fan-in and fan-out, the number of such pairs, and hence the number of polygons, is proportional to the number of nodes in the circuit.

From the perspective of a numerical analyst, the engineer's intuition means that a full dimension polyhedral representation of the reachable region may provide unneeded freedom in its ability to represent constraints between every possible combination of variables. In the same way that many matrices encountered in practice contain interaction between only limited sets of variables, in many ODE systems each variable only directly influences a small number of others. Dense storage and manipulation of sparse matrices is wasteful; similarly, representing the reachable state space as a full dimensional polyhedron may be exponentially extravagant.

Finally, there are algorithmic advantages to using projections. The existence of a sound method for computing the evolution of bounding polyhedra represented in this manner is key to verification. In addition, all geometric operations take place in two dimensions where there are many results

and algorithms available from computational geometry [PS85]. Lastly, it is relatively easy to compute the convex hull of a polygon, thus producing a containing approximation of that polygon in the form of a linear program.

Of course, there are many polyhedra that cannot be exactly represented by this approach. First, indentations on the surface of an object can not be represented; likewise, many perforated objects and knot-like objects can only be approximated. We require that the projections are orthogonal; therefore, edges formed by the intersection of projections must be at right-angles. Further experimentation is needed to determine the significance of these limitations when analysing circuits modeled by ODEs.

Reachability

Let $Q(\mathcal{P}_0)$ be a polyhedron, and let $\dot{x} = Ax$ be a linear model for a system. Given a monotonically increasing sequence of times, $t_1 \dots t_k$, we will construct a sequence of polyhedra $Q(\mathcal{P}_1) \dots Q(\mathcal{P}_k)$ such that trajectories that start in $Q(\mathcal{P}_0)$ at time $t = 0$ are contained in $Q(\mathcal{P}_i)$ at time $t = t_i$. Our approach is based on three observations, which we justify below. First, it is sufficient to consider trajectories emanating from the faces of $Q(\mathcal{P}_0)$, as these will define the faces of the polyhedron at later times. Second, for each edge e of a projection polygon, it is straightforward to construct a convex containing approximation for $face(e)$. Third, the method described in section 3.2 can be used to determine reachability from this convex approximation.

Because Ax is locally bounded, trajectories are continuous and cannot cross. Therefore, trajectories starting on a face of the polyhedron provide bounds for trajectories starting in the interior.

To construct a convex approximation for $face(e)$ let

$$Z(\mathcal{P}) = \bigcap_{P \in \mathcal{P}} \text{prism}(\text{ConvexHull}(P)) \quad (15)$$

It is straightforward to show that $\text{ConvexHull}(Q(\mathcal{P})) \subseteq Z(\mathcal{P})$, and $\text{ConvexHull}(face(e, \mathcal{P})) \subseteq \text{ConvexHull}(Q(\mathcal{P})) \cap \text{prism}(e)$. Therefore,

$$\text{ConvexHull}(face(e, \mathcal{P})) \subseteq Z(\mathcal{P}) \cap \text{prism}(e) \quad (16)$$

Given \mathcal{P} , a linear program for $Z(\mathcal{P})$ can be constructed by computing the convex hull for each polygon in \mathcal{P} and taking the conjunction of their constraints. Each polygon is two dimensional, allowing efficient (i.e. $O(n \log n)$) algorithms to be used. Once $Z(\mathcal{P})$ is calculated, it is easily extended to produce $Z(\mathcal{P}) \cap \text{prism}(e)$ for each edge. This provides our convex approximation of $face(e, \mathcal{P})$.

The method described above allows us to construct a $d - 1$ dimensional convex approximation for each face of $Z(\mathcal{P}_0)$. The reachable space from each face can then be computed by the techniques given in section 3.2. The boundary of the region reachable from $Z(\mathcal{P}_0)$ is contained in the union of the regions reachable from the faces.

In order for the same algorithms to be used for the next time step, we would like to compute a containing approximation of this boundary as a series of projections—describing the new boundary in the same way that $Z(\mathcal{P}_0)$ was described. Given a linear program for a face, the projection of that face onto a plane can be computed by finding an extremal vertex of the projection, and tracing the rest of the vertices with a series of pivots (see [AF92]). Because there may be an exponentially large number of vertices in this projection, such an approach may be slow. To avoid tracing too many vertices, extremal vertices can be computed for a fixed set of directions, and edges associated with these vertices joined to produce a containing approximation of the projection. Regardless of the method chosen to compute the projections, an object that contains everything reachable from $Q(\mathcal{P}_0)$ can be constructed by filling in the projection polygons (another straightforward operation).

An unattractive feature of this approach is that the reachable polyhedron for each face must be projected onto *all* planes used for the original projection polygons. Intuitively, this is because with a linear model, we can calculate the exact image of the convex approximations of the face for arbitrarily large times. During such an extended time interval, the polyhedron can rotate, and any face can become an extremal face for any projection.

3.4 Non-linear models and non-convex polyhedra

We extend the methods of the previous section to non-linear models in three steps. First, we will approximate the non-linear model by a linear model and a correction term. Second, we show how this correction term can be described as an non-determinate function of time, allowing the non-linear ODE to be approximated by a first order linear differential equation with an non-determinate nonhomogeneity. Finally, by bounding the solutions of the nonhomogenous system, we obtain a containing approximation of solutions to the original non-linear system.

Because the method from section 3.3 considers each face separately, we focus on the problem of finding the points reachable in time Δt from a point in $face(e)$ for some edge e , for a model whose derivative function has an L2 norm bounded by $\|\dot{x}\|_{\max}$. In determining the region reachable from $face(e)$, only points in $bloat(face(e), (\Delta t)\|\dot{x}\|_{\max})$ need to be considered. The derivation of the linear approximation and correction term is handled by the model—in other words, we leave it to the ingenuity of the programmer. When the model is evaluated, $bloat(face(e), (\Delta t)\|\dot{x}\|_{\max})$ is available as a linear program, so linear bounds can be readily obtained describing the region in which the approximation and correction must be valid.

As an example, consider the transistor model presented in equation 2 with $V_t = 0.5$. For a particular bloated face, assume $1.2 \leq V_{gs} \leq 1.6$ and $2.4 \leq V_{ts} \leq 3.1$. Then, everywhere in this region $i_{ds} = G(V_{gs} - V_t)^2$. Linearizing about the mid-point of the region and choosing an additive constant to minimize the worst-case absolute value of the error, we get $i_{ds} =$

$G(1.8V_{gs} - 1.69) \pm \epsilon(v_{gs}, v_{ds})$, where $\epsilon(v_{gs}, v_{ds}) \in [-0.02, 0.02]$. Similar techniques apply when the feasible region includes more or other modes of the transistor's operation.

Linear models can also be computed for input signals that are described using annuli (recall figure 2). As for the transistor model, the input signal model queries the linear program for the bloated face to determine upper and lower bounds for the value of the signal. For any given value of the signal, the annulus specifies upper and lower bounds for its time derivatives. From this description, a linear model with an error term can be computed. For such signals, the error term can be quite large; especially when the signal can be in the first (logical low) or third (logical high) regions of the annulus.

The non-linear correction term is a function of the state of a trajectory:

$$\dot{x} = Ax + \epsilon(x) \quad (17)$$

The model provides bounds on $\epsilon(x)$; thus, we write $\epsilon(x) \in E$ for some $E \subseteq \mathbb{R}^d$. For any particular trajectory, the correction term can be understood as a function of time, and we write

$$\dot{x} = Ax + \xi(t) \quad (18)$$

By computing the set of points reachable by trajectories for all functions ξ with $\xi(t) \in E$, we obtain a containing approximation for the original, non-linear system.

Equation 18 is a linear, nonhomogeneous, first-order differential equation. Such equations have a closed form solution [Apo67], namely:

$$x(t) = e^{tA}x(0) + e^{tA} \int_0^{\Delta t} e^{-uA} \xi(u) du \quad (19)$$

The $e^{tA}x(0)$ term is the solution to the linear approximation and the $e^{tA} \int_0^{\Delta t} e^{-uA} \xi(u) du$ term is the perturbation arising due to the non-linear correction in the model. A bound on the contribution of this correction term is computed next.

We assume that A has a full rank set of eigenvectors. If not, A can be perturbed slightly so as to satisfy this condition, and the perturbation can be reflected by slightly enlarging the correction term. Now, A can be diagonalized [HS74]; thus $e^{-tA} = D^{-1}e^{-tA^\dagger}D$, where D is the diagonalizing matrix, and A^\dagger is diagonal. The elements of e^{-tA^\dagger} (also a diagonal matrix) can be readily bounded for all $t \in [0, \Delta t]$. Using standard optimization techniques [PS82], a linear program can be constructed that is a containing approximation for the values of $e^{tA} \int_0^{\Delta t} e^{-uA} \xi(u) du$.

The previous paragraph provides a mathematically rigorous way to bound the contribution to trajectories of the non-linear component of the model. We expect that it would be impractical to implement this method due to its reliance on diagonalizing A —a procedure that is both time-consuming and

numerically sensitive. Instead, we plan to sample e^{-uA} for several values of $u \in [0, \Delta t]$ using a numerical approximation such as an integration algorithm. From these samples, approximate bounds on the non-linear contribution can be found. Just as with the mathematically rigorous approach, these bounds can be expressed as a linear program.

Using one of the methods in the previous two paragraphs, a containing approximation in linear program form for $e^{tA} \int_{u=0}^{\Delta t} e^{-uA} \xi(u)$ can be constructed. Section 3.3 built a linear program containing the values of $e^{tA} \text{face}(e)$. For reasons that will be explained shortly, we will instead use a linear program that contains the values of $e^{tA} \text{face}'(e)$, where

$$\begin{aligned} \text{face}'(e, (\Delta t) \|\dot{x}\|_{\max}) &= \text{bloat}(Z(\mathcal{P}), \|\dot{x}\|_{\max}) \cap \text{prism}(\text{extend}(e, (\Delta t) \|\dot{x}\|_{\max})) \\ \text{extend}(e, (\Delta t) \|\dot{x}\|_{\max}) &= e \text{ with end points extended outward by } (\Delta t) \|\dot{x}\|_{\max} \end{aligned}$$

Note that $\text{face}(e) \subseteq \text{face}'(e)$. A containing approximation for the sum of $e^{tA} \int_{u=0}^{\Delta t} e^{-uA} \xi(u)$ and $e^{tA} \text{face}'(e)$ can also be described by a linear program, and we can approximate the boundary of the reachable space at time Δt as the union of these linear programs for each face.

The methods described in this section rely on representation of the reachable space by a collection of two dimensional projections. For example, we use an approximation of the convex hull of the reachable space which is derived from the convex hulls of the projections. Furthermore, we need to know the endpoints of each edge when creating the convex approximation of the corresponding face. Finding the endpoints is straightforward when they are defined by segment intersections in a plane. Therefore, each integration step must end by computing projection polygons for the new reachable space object.

The technique described in section 3.3—projecting the convex hull for each transformed face onto each projection plane—could be applied here as well. For the methods described in this section, it is only necessary to project each transformed face back to the projection plane for its original edge. Let e be an edge of polygon P and e' be an adjacent edge of another polygon. Then e and e' are orthogonal. Also note that all points of $\text{face}(e')$ lie on the inside of $\text{face}'(e, (\Delta t) \|\dot{x}\|_{\max})$. Therefore, all trajectories starting from $\text{face}(e')$ remain on the inside of $\text{face}'(e)$ at the end of the time step. Thus, the projection of the boundary of the polyhedron into the plane of P is completely determined by the projection of the faces arising from edges in P at the beginning of the time step.

4 Conclusion

Many verification problems can be formulated as questions of reachability. With a circuit modeled by a system of ordinary differential equations, the

reachability problem can be formulated as: “given an initial region Q_0 and an ending time t_f (possibly $+\infty$), find a region Q such that all trajectories starting in Q_0 at time $t = 0$ remain in Q at least until time $t = t_f$.”

We have addressed this problem for three classes of models and regions. First considering linear models with convex regions, we showed how the region reachable at a future time can be computed exactly. Furthermore, a containing approximation for points reachable through all times up until that future time can be computed with a simple trade-off between effort and accuracy. We note that the HYTECH tool [HH95] represents reachable regions as a union of convex polyhedra, and it is possible that the techniques presented there could be applied in this first context.

Because models with non-linearities do not preserve the convexity of regions, it was next necessary to identify an efficient representation for non-convex polyhedra. For our purposes, projection polyhedra—where an object is represented by its projection onto two dimensional subspaces—provide such a representation, allowing us to apply efficient algorithms from computational geometry in two-dimensions to our higher dimensional problems.

Finally, we addressed the analysis of non-linear systems, by approximating the non-linear model by a linear term and a non-linear correction. The correction can be kept small by computing separate such models for each face of the reachable space, and can be approximated by a non-determinant “error” function of bounded magnitude. This construction allowed us to convert a non-linear model into a linear nonhomogenous differential equation, which can be solved analytically, and such solutions allow us to bound the reachable space.

The analysis presented in this paper shows that ideas from computational geometry, dynamical systems, formal methods, linear algebra, and numerical computation can all contribute to the verification of systems with ODE models. The authors are currently implementing a tool to demonstrate these techniques.

Acknowledgements

We appreciate an extended e-mail discussion with Oded Maler and Thao Dang on reachability with continuous models. Jack Snoeyink and Danny Chen have guided us about what is and what is not feasible in computational geometry.

References

- [AF92] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete Computational Geometry*, 8:295–313, 1992.
- [Apo67] Thomas M. Apostle. *Calculus*, volume 1. John Wiley and Sons, Inc., New York, second edition, 1967.

- [Bak90] H.B. Bakoglu. *Circuits, Interconnections, and Packaging for VLSI*. Addison-Wesley, 1990.
- [Bro89] R. W. Brockett. Smooth dynamical systems which realize arithmetical and logical operations. In Hendrik Nijmeijer and Johannes M. Schumacher, editors, *Three Decades of Mathematical Systems Theory: A Collection of Surveys at the Occasion of the 50th Birthday of J. C. Willems*, volume 135 of *Lecture Notes in Control and Information Sciences*, pages 19–30. Springer, 1989.
- [GD85] Lance A. Glasser and Daniel W. Dobberpuhl. *The Design and Analysis of VLSI Circuits*. Addison-Wesley, 1985.
- [GM97] Mark R. Greenstreet and Ian Mitchell. Reachability with discrete and ODE models. In Michael Lemmon, editor, *Fifth International Hybrid System Workshop*, Notre Dame, September 1997.
- [Gre96] Mark R. Greenstreet. Verifying safety properties of differential equations. In *Proceedings of the 1996 Conference on Computer Aided Verification*, pages 277–287, New Brunswick, NJ, July 1996.
- [HH95] T.A. Henzinger and P.-H. Ho. HyTECH: The Cornell Hybrid Technology Tool. In P. Antsaklis, A. Nerode, W. Kohn, and S. Sastry, editors, *Hybrid Systems II*, Lecture Notes in Computer Science 999, pages 265–293. Springer-Verlag, 1995.
- [HS74] Morris W. Hirsch and Stephen Smale. *Differential Equations, Dynamical Systems, and Linear Algebra*. Academic Press, San Diego, CA, 1974.
- [Nag75] L.W. Nagel. SPICE2: a computer program to simulate semiconductor circuits. Technical Report ERL-M520, Electronics Research Laboratory, University of California, Berkeley, CA, May 1975.
- [PS82] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall, Englewood Cliffs, NJ, 1982.
- [PS85] Franco P. Preparata and Michael I. Shamos. *Computational Geometry: An Introduction*. Texts and Monographs in Computer Science. Springer, 1985.

Lyapunov Stability of Continuous-Valued Systems under the Supervision of Discrete-Event Transition Systems

Kevin X. He and Michael D. Lemmon

Dept. of Electrical Engineering

University of Notre Dame

Notre Dame, IN 46556, USA

(219)-631-8309

fax:(219)-631-4393

lemmon@maddog.ee.nd.edu

1 Introduction

This paper examines the Lyapunov stability of equilibrium points for switched control systems [Mor95]. A switched control system is a continuous-valued system whose control law is switched in a discontinuous manner as the system state evolves over a continuous-valued subset of \mathbb{R}^n . Of particular interest in this paper are switched systems in which the switching logic is generated by a discrete-event transition system that can be represented as either a finite automaton or bounded Petri net.

There are a variety of prior results identifying sufficient conditions for such systems to be Lyapunov stable at their equilibrium point. In [Pel91] and [Sav96] a single positive definite functional is found which is Lyapunov for all control systems in the collection. Multiple Lyapunov function approaches in [Bra94] and [Hou96] have been presented which should be applicable to a larger set of systems than the single Lyapunov function methods. In certain cases, where the switched systems are linear time invariant and the switching regions are defined by conic sectors, it has been suggested that candidate Lyapunov functionals can be numerically computed by finding feasible points of a linear matrix inequality [Pet96] [Ran97].

While these prior results have provided great insight into the Lyapunov stability of switched systems, previously published results do not discuss the role or structure of the switching law in any detail. In the case of the computational methods proposed in [Pet96] and [Ran97], disregard for the switching laws structure may lead to linear matrix inequalities (LMI's) that are larger than needed and hence provide an overly restrictive sufficient condition for switched system stability. These conditions are overly restrictive in that the resulting system can only tolerate very small disturbances. This paper examines the numerical question and asks what sort of information about the switching law can be used to significantly reduce the computational complexity and conservatism associated with finding candidate Lyapunov functions of switched systems. The principal result of this paper states that if the switching law can be represented as a discrete-event transition system such as a finite automaton or Petri net, then it suffices to examine live fundamental cycles of the directed graph associated with such structures to assess switched system stability. In particular, the results

and viewpoints suggested in this paper provide a way in which the traditional control theoretic methods cited above can be combined with results from computer science [Alu94] [Alu96] concerned with the behaviour of timed transition systems.

The remainder of the paper is organized as follows. In section 2, we first introduce a formal model for switched control systems which are supervised by a discrete-event transition system. Section 3 states recent results [Bra94] [Pet96] providing sufficient conditions for switched system stability using a multiple Lyapunov function approach. Section 4 motivates, states, and proves the paper's principal result. Section 5 presents two examples illustrating the value of using fundamental cycles in assessing switched system stability. Section 6 concludes with topics and directions for further study.

2 Problem Statement

Let $X \subset \mathbb{R}^n$ be a smooth n -dimensional manifold and let I be a finite set of N integers. Let Δ be a constant dimensional distribution,

$$\Delta = \text{span}\{f_1, \dots, f_N\} \quad (1)$$

where $f_i : X \rightarrow X$ for $i = 1, \dots, N$ are locally Lipschitz vectorfields over X . We let a **switched dynamical system** be described by the following set of equations.

$$\dot{x}(t) = f_{i(t)}(x(t)) \quad (2)$$

$$i(t) = q(x(t), i(t^-)) \quad (3)$$

where $x : \mathbb{R} \rightarrow X$, $i : \mathbb{R} \rightarrow I$, and $q : X \times I \rightarrow I$. $i(t^-)$ refers to the righthand limit of the function $i(t)$ at point t . In the sequel, we refer to each f_i as a **subsystem** of the switched system. The preceding model is similar to that used in [Tav87].

A **trajectory** of the switched system is the ordered pair, (x, i) , where $x : \mathbb{R} \rightarrow X$ and $i : \mathbb{R} \rightarrow I$ which **solves** the system equation. The value taken by the trajectory at time $t \in \mathbb{R}$ is denoted by $(x(t), i(t))$. We say that (x, i) **solves** the system equation if and only if the equations are satisfied almost everywhere by $x(t)$ and $i(t)$ for $t \in \mathbb{R}$. This paper does not treat questions concerned with the existence of solutions. In general, however, solutions (when they do exist) will not be unique due to the nondeterminism in the switching law.

Let (x, i) be the trajectory generated by a switched dynamical system. The set of **switching times**, Ω , of a trajectory (x, i) will be

$$\Omega = \left\{ t : \lim_{\tau \rightarrow t^+} i(\tau) \neq \lim_{\tau \rightarrow t^-} i(\tau) \right\} \quad (4)$$

The set of **switching events**, \mathcal{E} , of trajectory (x, i) is denoted as

$$\mathcal{E} = \left\{ (i, t) \in I \times \mathbb{R} : t \in \Omega, i = \lim_{\tau \rightarrow t^+} i(\tau) \right\} \quad (5)$$

We define the **timed projection** $P_t : \mathcal{E} \rightarrow \mathfrak{R}$ by the equation $P_t[(i, \tau)] = \tau$ and the **event projection**, $P_e : \mathcal{E} \rightarrow I$ by the equation $P_e[(j, \tau)] = j$.

The **switching sequence**, is a mapping $\lambda : Z \rightarrow \mathcal{E}$ such that

$$P_t[\lambda(n)] < P_t[\lambda(n+1)] \quad (6)$$

for all $n \in Z$. Suppose λ is a switching sequence. Let I^* be the set of all strings formed from I . We let $\lambda_e = P_e[\lambda] \in I^*$ and $\lambda_t = P_t[\lambda]$ denote the event and time projections of λ , respectively. Let the subsequence of times when system j is turned on and off be denoted as $\lambda_{t,j} \in I^*$. In other words,

$$\lambda_{t,j} = \lambda_t(n_1), \lambda_t(n_1 + 1), \dots, \lambda_t(n_k), \lambda_t(n_k + 1), \dots \quad (7)$$

where n_k is a subsequence of Z such that $P_e[\lambda(n_k)] = j$. Define the interval completion $I(\lambda_{t,j})$ as the set obtained by taking the union of all open intervals in which system j is active. In other words,

$$I(\lambda_{t,j}) = \bigcup_{k=1}^{\infty} (\lambda_t(n_k), \lambda_t(n_k + 1)) \quad (8)$$

Denote $E(\lambda_{t,j})$ as a subsequence of $\lambda_{t,j}$ when the subsystem j is turned on. In other words,

$$E(\lambda_{t,j}) = \lambda_t(n_1), \lambda_t(n_2), \dots, \lambda_t(n_k), \dots \quad (9)$$

The preceding model of a switched system assumes a very general switching function, q . To obtain more precise results, however, we need to specify the nature of the switching function. A common choice is to associate a **discrete-event** transition system such as a finite automaton or Petri net with the switching system. In this paper we limit our scope to finite automata. An automaton is tied to the switched system by associating the vertices to the switched system's subsystems and by associating the arcs with *switching sets* called *guards*. The timed automaton [Alu94] and hybrid automaton [Alu96] provide tangible examples of this approach. In this paper we begin by considering a discrete-event transition system that is represented by a finite automaton, (V, A) .

A **finite automaton** associated with the switched system is the directed graph (V, A) where $V = I$ is a set of vertices and $A \subset V \times V$ is a set of directed arcs. By definition, the automaton associates a subsystem f_i with each vertex of the (V, A) . We define the **guard**, Ω_{ij} of arc $(i, j) \in A$ as

$$\Omega_{ij} = \{x \in X : j = q(x, i)\} \quad (10)$$

The ordered pair (i, j) is an arc of A if and only if $\Omega_{ij} \neq \emptyset$. The guard therefore represents a subset of the switched system's state space in which a switch can occur. The guard set Ω_{ii} will sometimes be denoted as Ω_i and represents the set in which subsystem f_i remains active.

The preceding paragraph characterized the switching logic by a finite automaton (V, A) . It is straightforward to generalize this approach to consider more complex switching logics. In particular, let's consider how this might be

done for a switching logic generated by a Petri net. A Petri net is represented by a directed graph (V, A) where the vertex set consists of two types of vertices, **places**, P , and **transitions**, T . The vertex set, therefore, takes the form $P \times T = V$. We associate this directed graph structure with the switched system by letting $P = I$. We therefore associate a subsystem with each place of the Petri net. The guards, Ω_{ij} , are associated with the transition $t \in T$ which connect the i th and j th places of the network. Petri nets provide natural structures for modeling concurrency and synchronization in parallel systems. In general, a Petri net can provide a more expressive characterization of a system's switching logic than can be provided by a finite automaton.

Let (x, i) be the trajectory generated by a switched dynamical system. The trajectory is said to be **deadlock free** if the event projection of the switching sequence $P_e[\lambda]$ is not finite. We say that the trajectory is **live** if the event projection of the switching sequence $P_e[\lambda]$ contains an infinite number of each index, $i \in I$. In other words any subsystem can be switched an infinite number of times in a switching sequence. We say that the trajectory is **nonZeno** if the timed projection of the switching sequence $P_t[\lambda]$ satisfies

$$\sum_{n=1}^{\infty} P_t[\lambda(n)] > \infty \quad (11)$$

We say that the switched system is live, deadlock free, or nonZeno if all of its trajectories are live, deadlock free, or nonZeno, respectively.

An important issue which is not addressed in this paper concerns necessary and sufficient conditions for a switched system to be live, deadlock free, or nonZeno. In this paper, we assume that the switched system is live and nonZeno.

3 Prior Results

This section briefly discusses prior results on switched system stability. Let (x, i) be any trajectory generated by the switched dynamical system. Assume that $f_i(0) = 0$ for all $f_i \in \Delta$. The equilibrium point $x = 0$ is said to be **stable in the sense of Lyapunov** if and only if for all $\epsilon > 0$ there exists $\delta > 0$ such that $\|x(t_0)\| < \delta$ implies $\|x(t)\| < \epsilon$ for all $t \geq t_0$.

In the following we will denote the open ball of radius r centered at the origin as

$$B(r) = \{x \in \mathbb{R}^n : \|x\| < r\} \quad (12)$$

The sphere, $S(r)$, of radius r centered at the origin is the set

$$S(r) = \{x \in \mathbb{R}^n : \|x\| = r\} \quad (13)$$

Let λ be a switching sequence for a switched dynamical system where λ_t is its time projection. we say that a continuously differentiable function $V : \mathbb{R}^n \rightarrow \mathbb{R}^+$ is **Lyapunov-like function** over sequence λ_t if and only if $\dot{V}(x(t)) \leq 0$ for all $t \in \mathcal{I}(\lambda_t)$ and V is monotonically nonincreasing on $E(\lambda_t)$. Using this definition of

a Lyapunov like function, the following sufficient condition for Lyapunov stability was proven in [Bra94]. The proof uses standard techniques employed in proving Lyapunov stability for nonautonomous systems. A significant generalization of this result will be found in [Hou96].

Theorem 1. *Suppose we have candidate Lyapunov functions V_j ($j \in I$) and suppose that the switched system is nonZeno and satisfies $f_i(0) = 0$ for all $j \in I$. If V_j is a Lyapunov like function for switching sequence $\lambda_{t,j}$ for all $j \in I$. then the equilibrium point $x = 0$ of the switched system is stable in the sense of Lyapunov.*

The preceding theorem provides a sufficient condition for Lyapunov stability of switched systems. The condition requires that a set of Lyapunov like functions be determined for all possible switching sequences λ that can be generated by the system. The determination of Lyapunov like functions may not be possible in general. For switched systems in which each subsystem is a linear time invariant system and the guard sets are represented by conic sectors in \mathbb{R}^n , a method for determining the Lyapunov like functions was presented in [Pet96] and [Ran97]. Assume that each subsystem can be written as

$$\dot{x}(t) = A_i x(t) \quad (14)$$

where $A_i \in \mathbb{R}^{n \times n}$ and $i \in I$. Assume that the guard sets can be bounded by conic sectors parameterized by symmetric matrices Q_{ij} . In other words, consider sets,

$$\Omega_{ij} \subseteq \{x \in \mathbb{R}^n | x' Q_{ij} x \leq 0\} \quad (15)$$

Ω_{ii} represents the set in which the i th subsystem is free to operate and Ω_{ij} (where $i \neq j$) denotes the guard set for the transition between the i th and j th vertices. If we can find real matrices, $P_i = P_i' > 0$ for all $i \in I$ and real constants $\alpha_i > 0$ and $\alpha_{ij} > 0$ such that

$$A_i' P_i + P_i A_i + \alpha_i Q_{ii} \leq 0 \quad (16)$$

$$P_i - P_j + \alpha_{ij} Q_{ij} \leq 0, \quad (17)$$

then the functionals, $V_j = x' P_j x$ are Lyapunov like functions of the switched system. This particular condition is more restrictive than that formulated in [Bra94]. But it can be readily reformulated as a linear matrix inequality (LMI) which can be solved using interior-point methods for convex optimization.

4 Main Result

The sufficient conditions presented in [Bra94] [Hou96] and used in [Pet96] [Ran97] to compute candidate Lyapunov functionals provide an approach for testing switched system stability. These methods, however, do not explicitly account for the structure of the switching logic. For example, the stability theorems in

[Bra94] [Hou96] require that V_j be Lyapunov like for all possible switching sequences. These papers place no assumptions on the nature of the switching laws. The computational methods demonstrated in [Pet96] assume no structure on the switching logic and therefore consider the worst case switching law in which every possible switch has to be considered. Neglecting the structure of the switching law can result in an extremely high dimensional linear matrix inequality which may be more restrictive than it needs to be.

In this section, we present and prove a result which shows that when the switching logic can be characterized by a finite automaton, then we only need to search for Lyapunov like functions over a restricted set of **fundamental cycles** in the finite automaton. Essentially, the following result shows that rather than having to examine whether a set of candidate functions are Lyapunov like for all possible switching sequences, we only need consider whether the candidate functions are Lyapunov like over a potentially smaller sized set of fundamental cycles.

Let the directed graph (V, A) have $n + 1$ vertices, i_0, i_1, \dots, i_n . The sequence of arcs

$$(i_0, i_1), (i_1, i_2), \dots, (i_{n-1}, i_n) \quad (18)$$

is called a path of length n . A **cycle** of a directed graph is any path such that $i_0 = i_n$. A cycle of length n

$$(i_0, i_1), (i_1, i_2), \dots, (i_{n-1}, i_0) \quad (19)$$

is said to be **fundamental** if $i_j \neq i_k$ for all j, k not equal to zero or n and for all $j \neq k$. The following results are basic facts from graph theory. In any fundamental cycle, any two vertices are connected by one and only one path. An arc of a directed graph that is in a cycle is also in a fundamental cycle. For any cycle, C , in a directed graph, there exists a set of fundamental cycles C_1, C_2, \dots, C_N such that

$$\text{Arcs}(C) = \bigcup_i^N \text{Arcs}(C_i) \quad (20)$$

Finally, the fundamental cycles of a directed graph can be determined in polynomial time by constructing a minimal spanning tree for the graph. Note that the fundamental cycles of a graph are non-unique.

To state and prove the main result of this paper, we first need to establish some facts about fundamental cycles generated by live switched systems. The first principal lemma is a result saying that any event sequence generated by a switched system can be constructed by recursively inserting fundamental cycles into a legal switching sequence. We then introduce a sufficient condition for a fundamental cycle to be **uniformly bounded** with respect to time. These two results are then combined to establish the Lyapunov stability of the entire switched system.

Lemma 2. *In the automaton associated with a live switched system, every arc is in at least one fundamental cycle*

Proof: Let (V, A) denote the finite automaton associated with a switched system. Assume that there exists an arc $(i, j) \in A$ which is not in any cycle of (V, A) . Therefore, once we go through arc (i, j) then there is no path back to vertex $i \in V$. Therefore in any switching sequence λ that contains arc (i, j) the number of times when vertex i is reached will be reached is finite which contradicts the definition of a live transition system. Therefore every arc of a live automaton is in a cycle. Furthermore from the fundamental results about cycles in directed graphs, we know that every arc is in at least one fundamental cycle, so the the lemma is proven. •

In the sequel, we will say an arc is a **live arc** if it is in a fundamental cycle.

Lemma 3. *Any switching sequence λ generated by a live switched system can be decomposed as*

$$\lambda_e = \sigma_1 \sigma_2 \sigma_3 \quad (21)$$

where σ_1 is a prefix of λ_e , σ_3 is a suffix of λ_e , and σ_2 is a fundamental cycle of the switched system's automaton.

Proof: Assuming there exists a switching sequence λ with event projection λ_e such that the decomposition doesn't exist. This means that there is no substring in λ_e which is a fundamental cycle. But from the definition of a live switched system, we know that every arc must be in a cycle. Let i_1 be the vertex where such a cycle starts. If the cycle is fundamental, then we have a contradiction and the proof is finished. But if the cycle is not fundamental, then there is a vertex i_2 which is crossed more than once in the cycle. Consider the cycle starting from i_2 . Either this cycle is fundamental, or not. If not, then we can repeat the above argument to find a smaller cycle within this one. However, because the automaton is finite, this recursion has to terminate in a fundamental cycle. We therefore have a contradiction and the lemma is proven. •

Proposition 4. *Given a switching sequence λ generated by a live switched system, let $\Lambda : Z \rightarrow I^*$ be a sequence of sequences in I^* constructed by the recursive procedure:*

1. $\Lambda[0]$ is a fundamental cycle C_0
2. $\Lambda[n] = \sigma_1 C_n \sigma_2$ where $\sigma_1 \sigma_2 = \Lambda[n-1]$ and C_n is a fundamental cycle.

Then there exists a set of C_i such that $\Lambda[n]$ is a prefix of λ for all n .

Proof: From lemma 3 we know that any switching sequence can be decomposed to $\sigma_1 \sigma_2 \sigma_3$ where σ_2 is a fundamental cycle. Note that if we pull out σ_2 from the switching sequence, then $\sigma_1 \sigma_3$ is still a possible switching sequence. We can now decompose the resulting sequence $\sigma_1 \sigma_3$ using lemma 3 to pull out another fundamental cycle of the automaton. Since the switching sequence is countable, we can repeat this process to pull out a countable sequence of fundamental cycles. This sequence is the set of C_i referred to in the above proposition.

•

A given sequence of events can be generated in various ways by a switched system. What we'd like to do is ensure that the cycle is well-behaved in some appropriate sense. In particular, we'll require that the continuous-state trajectory over the cycle is uniformly bounded with respect to time. The following lemma provides sufficient conditions for the system to be uniformly bounded.

Lemma 5. *Let λ_e be any cycle generated by the live switched system consisting of events*

$$\lambda_e = j_1, \dots, j_K \quad (22)$$

where $j_{K+1} = j_1$ with switching times

$$t_0, t_1, \dots, t_K \quad (23)$$

So that t_i is the time when the i th system is switched off and the $i + 1$ st system is switched on.

If there exist a set of continuously differentiable functions $V_j : \mathbb{R}^n \rightarrow \mathbb{R}$ for $j \in I$ such that $\dot{V}_j(x(t)) \leq 0$ for all $t \in [t_{j-1}, t_j]$, then for any $\epsilon > 0$ there $\delta(\epsilon) > 0$ such that for all $\|x(t_0)\| < \delta(\epsilon)$, $\|x(t)\| < \epsilon$ for all $t \in [t_0, t_K]$.

Proof: Consider an arbitrary $\epsilon > 0$ and let

$$\beta_K = \min_{x \in S(\epsilon)} V_{j_K}(x) \quad (24)$$

Define the closed set,

$$\Omega_K = \{x \in B(\epsilon) : V_{j_K}(x) \leq \beta_K\} \quad (25)$$

Choose ρ_K such that for all $x \in B(\rho_K)$, we have $V_{j_K}(x) < \beta_K$. We now define

$$\beta_{K-1} = \min_{x \in S(\rho_K)} V_{j_{K-1}}(x) \quad (26)$$

and introduce the closed set,

$$\Omega_{K-1} = \{x \in B(\rho_K) : V_{j_{K-1}}(x) \leq \beta_{K-1}\} \quad (27)$$

Choose ρ_{K-1} as was stated above and continue this process to construct a monotone sequence of sets

$$\Omega_1 \subseteq \Omega_2 \subseteq \dots \subseteq \Omega_{K-1} \subseteq \Omega_K \quad (28)$$

Note that Ω_j is invariant with respect to subsystem f_j because of the condition on \dot{V}_j . Therefore, we expect that if we start in $B(\rho_0)$, we should stay in set $B(\epsilon)$, which is sufficient to establish the lemma's conclusion. •

A cycle for which such functionals can be found will be said to be **uniformly bounded**. We now state and prove the main result of this section. This result uses the preceding proposition to show by induction that each of the sequences in the supersequence of lemma 3 is uniformly bounded if each fundamental cycle is uniformly bounded.

Theorem 6. Consider a live nonZeno switched system where $f_j(0) = 0$ for all $j \in I$. Let λ be a switching sequence generated by the system. Let μ denote a subsequence of contiguous switches in λ such that $P_e[\mu]$ is a fundamental cycle of the system's automaton. Let $\bar{\mu}$ denote the infinite sequence formed by concatenation of μ with itself.

If there exist a set of continuously differentiable functions $V_j : \mathbb{R}^n \rightarrow \mathbb{R}$ which are Lyapunov like over sequence $\bar{\mu}_{t,j}$ for all $j \in I$, then the system is stable in the sense of Lyapunov.

Proof: From our earlier lemma, we know that any switching sequence can be constructed by inserting fundamental cycles into a legal switching sequence. Let

$$A = \lambda[0], \lambda[1], \dots, \lambda[n], \dots \quad (29)$$

By definition $\lambda[0]$ is a fundamental cycle and under the theorem's hypothesis this is uniformly bounded.

Now assume that the sequence $A[n]$ is uniformly bounded. By assumption the fundamental cycle inserted into $A[n]$ is uniformly bounded. Note also, however, that since V_j is Lyapunov like we require that if $x(t_0) \in \Omega_1$, then it must return to that set. Hence the addition of the fundamental cycle does not change the boundedness of the original sequence $A[n]$. We can therefore conclude that $A[n+1]$ is uniformly bounded.

We now consider the limit as $n \rightarrow \infty$. Since the δ determined for uniform boundedness is independent of time, we can conclude that it holds for sequences of arbitrary length and hence the system is stable in the sense of Lyapunov. •

5 Example

In this section, we present some examples illustrating the application of the result in the preceding section to the computation of Lyapunov-like functionals using the LMI methods.

First, consider a live switched system whose automaton is shown in figure 1. Associated with each vertex is an LTI subsystem of the form

$$\dot{x} = A_i x \quad (30)$$

where $i = 1, 2, \dots, 6$. In addition to $A_i \in \mathbb{R}^{2 \times 2}$, we associate the "self-switching" set characterized by the symmetric matrix Q_i . Figure 1 shows the given automaton and the assumed matrices associated with each vertex. Each arc (i, j) in the automaton has a matrix Q_{ij} associated with it. The arcs are shown in figure 1 also.

From the automaton we can identify a set of four fundamental cycles. These fundamental cycles are obtained by determining a minimal spanning tree for the automaton's directed graph. This spanning tree is shown in figure 2 and the resulting fundamental cycles are $2 - 5 - 4$, $2 - 5 - 4 - 1$, $2 - 5 - 6$, $2 - 5 - 6 - 3$.

A similar set of inequalities can be formed for the other three cycles. To find the Lyapunov like functions, $V_i = x'P_i x$, we want to make sure that all fundamental cycles are stable, so we build a large LMI which includes all the matrix inequalities associated with the four fundamental cycles. For this example, there are a total of 15 matrix equations.

The feasibility of the 15 equation LMI can be readily checked using the LMI toolbox [LMI93]. As indicated before, the LMI's feasibility guarantees the Lyapunov stability of the switched system. In this example, however, the LMI is infeasible which means we cannot say whether the switched system is stable or not. Let's consider two ad hoc strategies for forcing such systems to be stable. The first strategy removes unstable fundamental cycles from the system. The second strategy determines a state feedback controller that stabilizes the system.

Consider the example system and check the feasibility of the LMI associated with each fundamental cycle. We discover that the LMI associated with cycle 2-5-6 is infeasible. Computer simulations show that cycle 2-5-6 is unstable. The first approach mentioned above will use a supervisory control to disable the transition from node 6 to node 2. The disabling of the transition essentially removes the unstable fundamental cycle from the system. With the unstable cycle disabled, we form a LMI for the three remaining fundamental cycles. In this example, the reduced LMI for the supervised system is feasible thereby indicating that the supervised switched system is Lyapunov stable. Simulation results for this example have verified this conclusion.

Alternatively, the second strategy determines a state feedback controller that stabilizes the unstable fundamental cycle. The LMI associated with the unstable cycle for the controlled system has the form,

$$(A_i + K_i)'P_i + P_i(A_i + K_i) + \alpha_i Q_i \leq 0 \quad i = 2, 5, 6$$

$$P_2 - P_6 + \alpha_{62} Q_{62} \leq 0$$

$$P_5 - P_2 + \alpha_{25} Q_{25} \leq 0$$

$$P_6 - P_5 + \alpha_{56} Q_{56} \leq 0$$

where $K_i \in \mathbb{R}^2$, $i = 2, 5, 6$ are stabilizing controllers to be determined. This is a nonlinear matrix inequality that can be transformed into a LMI by reparameterizing the feedback controller, K_i . Let $K_i = P_i^{-1}V_i$, then these inequalities become

$$A_i'P_i + P_iA_i + V_i' + V_i + \alpha_i Q_i \leq 0 \quad i = 2, 5, 6$$

The resulting matrix inequality is clearly linear. We combine the above LMIs for the unstable cycle with the LMIs for the other three fundamental cycles and use the LMI toolbox to check for a feasible solution. In this example, the resulting LMI is feasible and the solution obtained by the LMI toolbox determines the matrices P_i and the controller gain K_i which stabilize the system.

We now consider another example in which some transitions in the automaton are not live. Consider the switched system whose automaton is shown in figure 3.

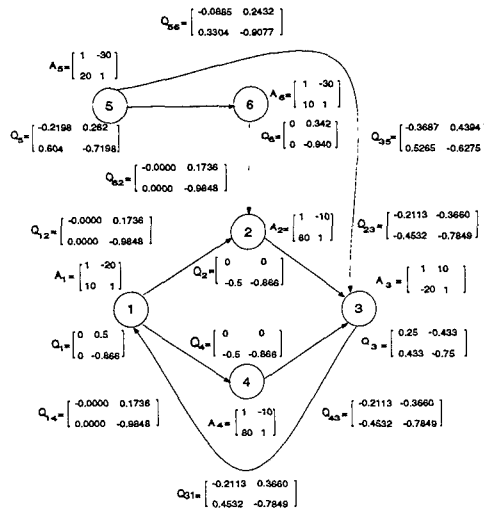


Fig. 3. The automaton of the switched system of example 2

From the automaton, we identify two fundamental cycles, namely, cycle $1-2-3$ and cycle $1-4-3$. We thus know that the system is not a live system, since arc $(5, 6)$, $(6, 2)$ and $(5, 3)$ are not in either of the fundamental cycles. One implication of our new result is that we can identify the "live part" of the system by identifying all the fundamental cycles in its automaton and all the live arcs associated with them. From lemma 2, we know that a live arc is guaranteed to appear infinite times in a switching sequence, whereas an arc which is not a live arc can appear only once in a switching sequence and therefore its appearance can not affect the stability of the switched system. We thus only need to consider all the live arcs in deciding stability of the whole switched system. In this example, we only need to establish LMI for the two fundamental cycles and check the existence of a set of Lyapunov-like functions V_j for $j = 1, 2, 3, 4$.

Using the same method as in the previous example, we find a total number of 9 LMIs for the two fundamental cycles. In comparison, if we had proceeded using the technique originally proposed in [Pet96], then we would need to build an LMI which accounted for all individual transitions that could possibly happen. If the automaton had N vertices and A arcs, then we would have at least $N + A$ equations in our linear matrix inequality. For our particular example, we would have a 14 equation LMI to solve.

The implication of increasing LMI size is that it represents an overly restrictive sufficient condition for system stability. In our case, we can see this quite easily by solving the 9 equation LMI obtained by examining the fundamental cycles of the system versus the 14 equation LMI obtained by using the methods in [Pet96]. The P matrices obtained in both cases for our example are shown in figure 4

In computing the first table, the LMI toolbox required 13326 flops to deter-

	original method	simplified method
P_1	$\begin{bmatrix} 0.0418 & -0.0084 \\ -0.0084 & 0.0677 \end{bmatrix}$	$\begin{bmatrix} 68.6535 & -12.4057 \\ -12.4057 & 113.0276 \end{bmatrix}$
P_2	$\begin{bmatrix} 0.0309 & -0.0020 \\ -0.0020 & 0.0064 \end{bmatrix}$	$\begin{bmatrix} 57.2575 & -2.5743 \\ -2.5743 & 11.0493 \end{bmatrix}$
P_3	$\begin{bmatrix} 0.0363 & 0.0106 \\ 0.0106 & 0.0200 \end{bmatrix}$	$\begin{bmatrix} 62.5312 & 22.6958 \\ 22.6958 & 35.5830 \end{bmatrix}$
P_4	$\begin{bmatrix} 0.0335 & -0.0015 \\ -0.0015 & 0.0050 \end{bmatrix}$	$\begin{bmatrix} 57.5544 & -1.9279 \\ -1.9279 & 8.2600 \end{bmatrix}$
P_5	$\begin{bmatrix} 0.0980 & -0.0028 \\ -0.0028 & 0.1315 \end{bmatrix}$	
P_6	$\begin{bmatrix} 0.0447 & -0.0068 \\ -0.0068 & 0.1218 \end{bmatrix}$	

(31)

Fig. 4. P matrices for example 2

mine the P matrices for the original method. The simplified method developed in this paper only required a total of 8665 flops. So our method clearly has a lower computational complexity than the original method of [Pet96]. More important than this, however, is the difference between the matrices. As can be clearly seen above, the singular values for the P matrices obtained from the simplified approach are around 50. For the original approach in [Pet96], however, these values are about .1. Since the singular value is a measure of how close the matrix is to being singular, this means that the original method was almost unable to determine the candidate Lyapunov functions. With minor changes in the Q matrices it is quite possible to generate examples in which the original method is unable to find the required P matrices, but our method would find such matrices. In addition, the size of the singular values for P provide an upper bound on the size of disturbance which the system can tolerate. The larger the singular value is, the larger this upper bound is. Clearly, the simplified method provides a largher upper bound on the disturbance that can be tolerated. So, the simplified method provides a less conservative assessment of the system's stability than the original method.

Remark: The use of fundamental cycles in assessing system stability is, in fact, more closely related to the results of [Hou96] rather than [Bra94]. The significance of the [Hou96] results, in our opinion, lies in the fact that one only need establish Lyapunov like behaviour over discrete switching times. In this regard, we should be able to use our results to determine potentially less restrictive stability conditions in which all Lyapunov-like functions need not always be monotone decreasing.

Remark: The LMI we constructed using the [Pet96] method actually takes into account all the possible switchings, i.e. all the arcs in the automaton to determine the system's stability, whereas our simplified method only consider all the live arcs, since we know that only the live arcs can affect the switched

system's stability. Therefore, it is obvious that our simplified method is less restrictive than the [Pet96] method.

Remark: The value of fundamental cycles go well beyond the immediate objective of saying a "yes" or "no" assessment of system stability. Fundamental cycles represent a basic characterization of the automaton's graph which can be very useful in analysis and synthesis. In particular, we believe it should be possible to use fundamental cycles to help decouple the LMI construction problem into a set of smaller problems. We also believe that the identification of "unstable" or "uncontrollable" fundamental cycles should provide a basis for introduction of supervisory control schemes in the switching law. The implication here is that the use of fundamental cycles in the qualitative analysis of switched systems potentially represents an important tool for the analysis and synthesis of switched systems. The stability analysis presented in this paper is only a simple example illustrating its potential use.

6 Future Work

This paper has presented a sufficient method for switched system stability which takes advantage of prior knowledge of the system's switching logic. In particular, it was shown that if the switching logic can be shown to be generated by a finite discrete-event transitions system such as a finite automaton or Petri net, then it suffices to determine Lyapunov-like functions only over the fundamental cycles of the state machine.

The preliminary results presented in this paper are encouraging and suggest several possible directions for future study. One future direction involves extending the concepts introduced here to study switching logics generated by Petri nets [Lem98]. The use of unfolding methods should allow the efficient identification of fundamental cycles in the Petri net's reachability tree, thereby providing a sufficient test for the stability of such systems. Another promising avenue of future study involves developing sufficient tests for uniform ultimate boundedness (bounded-amplitude) in switched systems. For important classes of systems, we can also formulate these sufficient conditions as matrix inequalities [Bet97] thereby allowing the efficient testing of switched system performance with respect to a specified ultimate bound. Examples of this approach will be also be found in [Lem98].

7 Acknowledgements

We gratefully acknowledge the partial financial support of the Army Research Office (DAAH04-95-1-0600) and the National Science Foundation (NSF-ECS95-31485).

References

- [Mor95] A.S. Morse, Control using logic based switching, In A. Isidori, Trends in Control, Springer-Verlag, Great Britain, 1995.
- [Pel91] P. Peleties and R. DeCarlo, Asymptotic Stability of m -switched systems using Lyapunov-like functins. In *Proceedings of the American Control Conference*, pp 1679-1684, Bost, MA, June 1991.
- [Sav96] A.V. Savkin, I.R. Petersen, E. Skafidas, and R.J. Evans, Robust control via controlled switching, In *Proceedings of CESA '96*, pages 1117-1122, Lille France, 1996
- [Bra94] M. Branicky, Stability of Switched and Hybrid Systems, In *Proceedings of the 33rd Conference on Decision and Control*, pp 3498-3503, Lake Buena Vista, FL, December 1994.
- [Hou96] L. Hou, A.N. Michel, and H. Ye, Stability Analysis of Switched Systems, In *Proceedings of the 35th Conference on Decision and Control*, pages 1208-1212, Kobe Japan, December 1996.
- [Pet96] S. Pettersson and B. Lennartson, Stability and Robustness for Hybrid Systems, In *Proceedings of the 35th Conference on Decision and Control*, Kobe Japan, December 1996.
- [Ran97] M. Johansson and A. Rantzer, Computation of Piecewise Quadratic Lyapunov Functions for Hybrid Systems, to appear in *IEEE Trans. of Automatic Control*, 1997
- [Alu94] R. Alur and D.L. Dill, A Theory of Timed Automata, *Theoretical Computer Science*, 126:183-235, 1994
- [Alu96] R. Alur, T. Henzinger, and P-H Ho., Automatic Symbolic Verification of Embedded Systems, *IEEE Transactions on Software Engineering*, 22:181-201, 1996.
- [Tav87] L. Tavernini, Differential automata and their discrete simulators, *Nonlinear Analysis, theory, methods, and applications*, 11(6):665-683, 1987.
- [Bra95] . S. Branicky, Studies in Hybrid Systems: Modeling, Analysis, and Control, Technical Report LIDS-TH-2304 of the Laboratory for Information and Decision Sciences, MIT, 1995.
- [Des95] A. Deshpande and P. Varaiya, Viable Control of Hybrid Systems, *Hybrid Systems II*, A. Nerode (ed.), LNCS Volume 999, Springer-Verlag, 1995.
- [Lem98] M.D. Lemmon, K. He, and C.J. Bett, Modeling Hybrid Control Systems using Programmable Timed Petri Nets, to appear in *L'Automatisation des Processus Mixtes (ADPM'98)*, Rheims France, March 19-20, 1998.
- [Bet97] C.J. Bett and M.D. Lemmon, Bounded Amplitude Control using Multiple Linear Agents, Technical Report ISIS-97-004, Dept. of Electrical Engineering, University of Notre Dame, Notre Dame, IN 1997. (revised version submitted to *Automatica*)
- [LMI93] Matlab Inc., "LMI Toolbox User Manual", 1993.

Reachability Verification for Hybrid Automata

Thomas A. Henzinger^{1*} and Vlad Rusu^{2**}

¹ EECS Department, University of California, Berkeley, CA
tah@eecs.berkeley.edu

² SRI International, Computer Science Laboratory, Menlo Park, CA
rusu@csl.sri.com

Abstract. We study the reachability problem for hybrid automata. Automatic approaches, which attempt to construct the reachable region by symbolic execution, often do not terminate. In these cases, we require the user to guess the reachable region, and we use a theorem prover (Pvs) to verify the guess. We classify hybrid automata according to the theory in which their reachable region can be defined finitely. This is the theory in which the prover needs to operate in order to verify the guess. The approach is interesting, because an appropriate guess can often be deduced by extrapolating from the first few steps of symbolic execution.

Keywords: hybrid automata, reachability verification, theorem proving.

1 Introduction

Hybrid automata are a specification and verification model for hybrid systems [ACH⁺95], systems that involve mixed continuous and discrete evolutions of variables. The problem that underlies the safety verification for hybrid automata is *reachability*: can an unsafe state be reached from an initial state by executing the system? The traditional approach to reachability attempts to compute the set of reachable states by successive approximation, starting from the set of initial states and repeatedly adding new reachable states. This computation can be automated and is guaranteed to converge in some special cases [KPSY93, AD94, ACH⁺95, HKPV95, RR96], for which the reachability problem is decidable. In general, however, this approach, which we call *reachability construction*, may not be automatable or may not converge.

It is for this reason that in this paper, we pursue a different approach, called *reachability verification*. In reachability verification, the user guesses the set of reachable states, and then a theorem prover is applied to verify the guess. A guess has the form of a logical formula, which is true exactly for the states that are guessed to be reachable. We classify hybrid automata as to what logical

* This research was supported in part by the ONR YIP award N00014-95-1-0520, by the NSF CAREER award CCR-9501708, by the NSF grant CCR-9504469, by the AFOSR contract F49620-93-1-0056, by the ARO MURI grant DAAH-04-96-1-0341, by the ARPA grant NAG2-892, and by the SRC contract 95-DC-324.036.

** Supported by Lavoisier grant of the French Foreign Affairs Ministry and by SRI.

theory suffices to define the set of reachable states. The formula to be guessed must lie in this theory, and the verification part amounts to a proof in this theory. Hence, the simpler the theory, the more constrained the guess and the easier the verification. In some cases—for example, the case of *additive-inductive* hybrid automata, where the set of reachable states is definable in a decidable subtheory of $(\mathbb{R}, \mathbb{N}, +, \leq)$ —the verification part is often completely automatic. The reachability verification approach is interesting because when successive approximation does not converge, a suitable guess can often be found by studying and extrapolating the first few iterations of successive approximation. In this way, some automatic heuristics can be developed to aid the guessing part.

The rest of the paper is organized as follows. In Section 2, we present the hybrid automaton model, the reachability construction method, and the reachability verification method. We restrict our attention to *linear* hybrid automata, for which reachability construction can be automated and has been implemented in verification tools such as HyTech [AHH96]. In Section 3, we classify linear hybrid automata according to the theory in which the set of reachable states is definable. For example, all linear hybrid automata for which reachability construction converges are *polyhedral*, as their reachable region can be defined in $(\mathbb{R}, +, \leq)$. We give examples of linear hybrid automata whose reachable regions are quite simple yet non-polyhedral (e.g., additive-inductive), as well as examples of linear hybrid automata whose reachable regions are quite complex (e.g., most naturally expressed using trigonometric functions). We also present a restricted subclass of additive-inductive automata for which the reachable region can be computed algorithmically, even though reachability construction does not necessarily terminate. Finally, in Section 4 we describe an embedding of hybrid automata into the theorem prover Pvs [ORR⁺96], and apply the reachability verification method to some well-known examples for which reachability construction fails.

2 Linear Hybrid Automata and Reachability Analysis

Hybrid automata [ACH⁺95] are finite automata enriched with a finite set of real-valued variables. In each location, the variables evolve continuously according to differential *activities*, as long as the location's *invariant* remains true; then, when a transition *guard* becomes true, the control may proceed to another location, and *reset* some of the variables to new values. We restrict our attention to a simple class of hybrid automata, allowing only straight-line activities and resets of variables to zero. More general feature can be approximated in the simpler framework, with additional locations, transitions, and variables [HHWT98].

Below (Figure 1) is an example of a linear hybrid automaton. It has the three locations s_1, s_2, s_3 and the three variables x, y, z . The automaton starts at location s_1 with variable x set to 0 and variables y, z set to 1, and control can remain at location s_1 while the invariant $x \leq y$ is true. Here, x increases with slope 1 ($\dot{x} = 1$) and y remains constant at 1 ($\dot{y} = 0$). Thus, control can stay at s_1 for at most 1 time unit, until x reaches 1. When this condition becomes

true, control leaves s_1 by taking a transition. Here, the only available transition is the one that leads to s_2 , which is enabled when $x = y$. Then, control goes to location s_2 , where x decreases ($\dot{x} = -1$), and stays there until x reaches 0. When this happens, the transition from s_2 to s_3 is enabled and control goes to s_3 , by assigning variable z to 0 in the process. The process continues likewise at location s_3 .

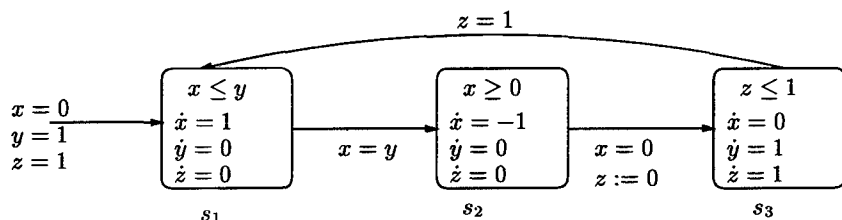


Figure 1. Example of a linear hybrid automaton

Syntax of linear hybrid automata. A convex linear predicate is a system of linear inequalities over given variables. A linear predicate is a finite disjunction of convex linear predicates. A linear hybrid automaton consists of the following elements:

- a finite set $X = \{x_1, x_2, \dots, x_n\}$ of variables;
- a finite set L of locations;
- a finite multiset of transitions $T \subseteq L \times L$;
- for each location $l \in L$:
 - an *invariant* $Inv(l)$, which is a convex linear predicate on the variables;
 - an *activity* $Act(l)$, which is a tuple of differentials laws (one law per variable) of the form $\dot{x} = A(l, x)$. Here, $A(l, x)$ is a rational constant, also called the *slope* of variable x at location l ;
 - an *initial condition* $Init(l)$, which is a convex linear predicate on the variables;
- for each transition $t \in T$:
 - a *guard* $Guard(t)$, which is a convex linear predicate on the variables;
 - a *reset* $Reset(t)$, which is a set of variables $Reset(t) \subseteq X$.

Semantics. The semantics of hybrid automata builds upon the following preliminary notions. A *valuation* is a function $v : X \rightarrow \mathbb{R}$ that associates a real number $v(x)$ to each variable $x \in X$. Given a variable valuation v and a linear predicate P over the variables, we say v satisfies P , written $P(v) = \text{true}$, if by replacing in P each variable x with its value $v(x)$, one obtains a true

statement. In particular, if valuation v satisfies the invariant of location l (respectively, the guard of transition t) we write $Inv(l)(v) = true$ (respectively, $Guard(t)(v) = true$). Given a valuation v and a subset $Y \subseteq X$ of variables, we write $v[Y := 0]$ for the valuation that assigns 0 to all variables in Y , and agrees with v on all variables in $X \setminus Y$. Given a valuation v , a location $l \in L$, and a non-negative real $\tau \in \mathbb{R}^+$, we write $v +_l \tau$ for the valuation that assigns to each variable x in X the value $v(x) + A(l, x) \cdot \tau$, where $A(l, x)$ is the slope of variable x at location l .

The semantic features of a hybrid automaton are the following:

- a *state* is a pair (l, v) , where l is a location and v a valuation such that $Inv(l)(v) = true$;
- for a non-negative real $\tau \in \mathbb{R}^+$, there is a *continuous step* of duration τ between two states (l, v) and (l, v') denoted $(l, v) \xrightarrow{\tau} (l, v')$, if $v' = v +_l \tau$;
- for a transition $a = (l, l') \in T$, there is a *discrete step* of label a between two states (l, v) and (l', v') denoted $(l, v) \xrightarrow{a} (l', v')$, if $Guard(t)(v) = true$ and $v' = v[Reset(t) := 0]$;
- a *run* is a finite sequence of continuous and discrete steps $(l_0, v_0) \rightarrow (l_1, v_1) \rightarrow \dots \rightarrow (l_m, v_m)$ such that the first state (l_0, v_0) is initial; i.e., v_0 satisfies the initial condition $Init(l_0)$.

A state is *reachable* if it coincides with the last state of a run. A *linear region* is a pair $\langle l, P \rangle$, where l is a location and P is a linear predicate on the automaton variables. A state (s, v) *satisfies* the linear region $\langle l, P \rangle$ if $s = l$ and v satisfies P . In this case we also say that the region $\langle l, P \rangle$ *contains* the state (s, v) . The *reachability problem* for linear hybrid automata is: given a linear hybrid automaton \mathcal{A} and a set \mathcal{R} of linear regions, is there a reachable state of \mathcal{A} that is contained in some region in \mathcal{R} . We discuss below two kinds of approaches to this problem.

Reachability construction [ACH⁺95]. This method performs a symbolic execution of the hybrid automaton. It consists in successively approximating the reachable region, starting from the initial region, and iterating *successor* operations until the computation converges. There are two kinds of successors.

The *continuous successor* of a region $\langle l, P \rangle$ is the region $\langle l, \tilde{P}_l \rangle$ that contains all the states that can be reached from states satisfying $\langle l, P \rangle$, by a single continuous step. The predicate \tilde{P}_l is obtained by *extension of P at location l* . Suppose P is a linear predicate over the variables x_1, \dots, x_n , and that variable x_i evolves in location l by the law $\dot{x}_i = k_i$ (for all $i \in \{1, \dots, n\}$); then, the extension of P at location l is described by the following predicate:

$$\tilde{P}_l = \exists \tau \geq 0. P(x_1 - k_1 \cdot \tau, \dots, x_n - k_n \cdot \tau) \quad (1)$$

It can be shown that the elimination of the existential quantifier in formula (1) can be performed, and it again produces a linear predicate in variables x_1, \dots, x_n : the continuous successor of a linear region is still a linear region.

The *discrete successor* of a linear region $\langle l, P \rangle$ by a transition $(l, l') \in T$ is the region $\langle l', \vec{P}_{(l, l')} \rangle$ that contains all the states that can be reached from states satisfying $\langle l, P \rangle$, by a single discrete step. The predicate $\vec{P}_{(l, l')}$ is obtained from P by *projection over transition* (l, l') . Suppose that P is a linear predicate over variables x_1, \dots, x_n , and transition (l, l') resets the variables $x_{i_1}, x_{i_2}, \dots, x_{i_p}$; then, the projection of P over transition (l, l') can be described by the following predicate:

$$\vec{P}_{(l, l')} = (x_{i_1} = 0 \wedge x_{i_2} = 0 \wedge \dots \wedge x_{i_p} = 0) \wedge \exists x_{i_1} \dots \exists x_{i_p} . P(x_1, \dots, x_n) \quad (2)$$

It can be shown that the elimination the existential quantifiers in formula (2) can be performed, and it again produces a linear predicate in variables x_1, \dots, x_n . Thus, the discrete successor of a linear region is still a linear region.

Reachability construction consists in iterating the following *Post* procedure:

Input: set A of linear regions.

Output: set B of linear regions, initially empty.

For each linear region $\langle l, P \rangle$ in the set A , for each transition (l, l') with origin l :

- let P_1 be the *intersection* of P with the *guard* of transition (l, l')
- let P_2 be the *projection* of P_1 over transition (l, l')
- let P_3 be the *intersection* of P_2 with the *invariant* of l'
- let P_4 be the *extension* of P_3 at state l'
- let P_5 be the *intersection* of P_4 with the *invariant* of l'
- add $\langle l', P_5 \rangle$ to set B .

We denote by $Post^k(I)$ the set of regions obtained by applying k times the *Post* operation to the set of initial regions $I = \{\langle l, Init(l) \wedge Inv(l) \rangle \mid l \in L\}$, and by $Post^*(I)$ the countably infinite union $\bigvee_{k \in \mathbb{N}} Post^k(I)$. This is also called the *reachable region*, and it represents all the reachable states of the hybrid automaton. Once $Post^*(I)$ is computed, the reachability problem for a set of linear regions \mathcal{R} can be solved by checking if the intersection $Post^*(I) \cap \mathcal{R}$ is non-empty.

We call *reachability construction* the process of computing the sequence $I, Post(I), Post^2(I) \dots$ of sets of regions. If, for some $k \in \mathbb{N}$, it is the case that $Post^{k+1}(I) \subseteq Post^k(I)$, then reachability construction terminates in finitely many steps, and $Post^*(I) = Post^k(I)$. This does not happen in general for linear hybrid automata [HKPV95]. Some subclasses for which reachability construction terminates have been identified, such as timed automata, initialized rectangular hybrid automata¹, and others [KPSY93, AD94, ACH⁺95, HKPV95, RR96]. For these classes, the reachability problem is decidable. Reachability construction is the procedure implemented in symbolic model-checking tools like HyTech [AHH96].

¹ For these classes, termination is achieved by slightly modifying the automaton.

Reachability verification. We define a new approach to the reachability problem, called the *reachability verification* method. This method can succeed in cases when reachability construction fails. Reachability verification consists of two steps: first, to *guess* the reachable region; second, to *verify* that the guess is correct. In many cases (some of which are presented in Sections 3 and 4), a suitable guess can be found using the simple heuristic described below, and the verification can be performed by induction, using a theorem prover.

It appears that when reachability construction does not terminate, the reachable region of a hybrid automaton can still behave in a regular manner. As an example, consider the hybrid automaton in Figure 1. By studying the reachability construction over a few iterations (performed in this situation by the tool HyTech), it can be seen that the reachable region is described by the following set of regions:

$$\begin{aligned} &\{ \langle s_1, \exists i \in \mathbb{N}. (i \geq 1 \wedge x \leq i \wedge y = i \wedge z = 1) \rangle, \\ &\langle s_2, \exists i \in \mathbb{N}. (i \geq 1 \wedge x \leq i \wedge y = i \wedge z = 1) \rangle, \\ &\langle s_3, \exists i \in \mathbb{N}. (i \geq 1 \wedge x = 0 \wedge y = z + i \wedge i \leq y \leq i + 1) \rangle \} \end{aligned} \quad (3)$$

The above expression involves a quantifier over a new natural-number variable i . Thus, a simple heuristic to guess the reachable region is to observe a few iterations of reachability construction, and to search for a reachable region of the form $\exists i_1 \in \mathbb{N} \dots \exists i_q \in \mathbb{N}. \mathcal{R}(i_1, \dots, i_q)$; that is, the reachable region involves some new natural-number variables i_1, \dots, i_q in addition to the automaton variables.

A typical situation is to guess a reachable region written using only one natural-number variable j , which represents the number of iterations of the *Post* procedure. In this case, we call the guessed region *directly inductive*, and proving that the guess is correct amounts to prove that for all $j \in \mathbb{N}$, $Post^j(I) = \mathcal{R}(j)$. This can be performed by induction using a theorem prover. In particular, we need to show the two following proof obligations: $\mathcal{R}(0) = \{Init(l)_l \wedge Inv(l) \mid l \in L\}$ for the base step, and $\forall j \in \mathbb{N}. Post(\mathcal{R}(j)) = \mathcal{R}(j + 1)$ for the induction step. As we shall see in Section 3, these proof obligations can often be discharged automatically.

In other situations the guessed region may not be directly inductive, but it can be made so by introducing new variables and constraints. For instance, the reachable region defined by expression (3) is not directly inductive, since the natural-number variable i does not represent the number of iterations. But this region becomes directly inductive by adding the constraints $j = 3i$, $j = 3i + 1$, and $j = 3i + 2$ respectively to the three regions in the set (3). That is, we define the sets of regions $\mathcal{R}(j) = \{ \langle s_1, \exists i \in \mathbb{N}. (j = 3i \wedge i \geq 1 \wedge x \leq i \wedge y = i \wedge z = 1) \rangle, \langle s_2, \exists i \in \mathbb{N}. (j = 3i + 1 \wedge i \geq 1 \wedge x \leq i \wedge y = i \wedge z = 1) \rangle, \langle s_3, \exists i \in \mathbb{N}. (j = 3i + 2 \wedge i \geq 1 \wedge x = 0 \wedge y = z + i \wedge i \leq y \leq i + 1) \rangle \}$ and now the “guess” $\exists j \in \mathbb{N}. \mathcal{R}(j)$ is directly inductive, with j representing the number of iterations.

Finally, even in situations when the guess is not (or cannot be transformed into) directly inductive, a useful approach is to prove that it is an *invariant* of the system. This can often be done automatically and it is often enough in practice for proving safety properties. We present sample proofs in Section 4.

We now give a classification of hybrid automata according to the theory in which their reachable region can be written finitely. The less expressive this theory, the less interactive theorem proving is needed for doing reachability verification.

3 Reachable Region Classification

The first class that we define contains in particular all the hybrid automata for which reachability construction terminates.

Definition 3.1 (polyhedral hybrid automata). A linear hybrid automaton is *polyhedral* if its reachable region can be expressed as a set of pairs $\{\langle l, P_l \rangle \mid l \in L\}$ such that for each location $l \in L$, P_l is a formula of the theory² $(\mathbb{R}, +, \leq)$. \square

We say a linear hybrid automaton is *finitely constructible* if its reachability construction terminates: i.e., for some $k \in \mathbb{N}$, $Post^*(I) = Post^k(I)$. While all finitely constructible hybrid automata are polyhedral, the converse is not true: it is easy to construct a hybrid automaton such that for all $k \in \mathbb{N}$, $Post^k(I)$ is the closed interval $[0, k]$; thus, the reachable region $Post^*(I)$ is the interval $[0, \infty)$, but reachability construction does not converge in finitely many steps. The class of finitely constructible hybrid automata includes the timed automata [AD94] and the initialized rectangular hybrid automata [HKPV95] (with some minor modifications to force the reachability construction to terminate) as well as some other restricted classes [KPSY93, RR96].

Definition 3.2 (additive-inductive hybrid automata). A linear hybrid automaton is *additive-inductive* if its reachable region can be expressed as a set of pairs $\{\langle l, P_l \rangle \mid l \in L\}$ such that for each location $l \in L$, P_l is a formula of the theory $(\mathbb{R}, \mathbb{N}, +, \leq)$ in which *all natural-number variables are outermost existentially quantified*. \square

For instance, the hybrid automaton in Figure 1 is additive-inductive: we have seen that its reachable region (3) involves the real variables x, y, z and the natural-number variable i , which is outermost existentially quantified.

Proposition 3.3. *The class of polyhedral hybrid automata is strictly included in the class of additive-inductive hybrid automata.*

Proof. The inclusion is obvious (since any formula of $(\mathbb{R}, +, \leq)$ is also a formula of $(\mathbb{R}, \mathbb{N}, +, \leq)$). Let us show that the inclusion is strict. For this, consider the hybrid automaton in Figure 1. We have seen that it is additive-inductive, and let us suppose it is finitely constructible, thus polyhedral by a previous observation. Then, formula (4) $\exists i \in \mathbb{N}. (i \geq 1 \wedge x \leq i \wedge y = i \wedge z = 1)$ can be also expressed in the theory $(\mathbb{R}, +, \leq)$; that is, the set of triples (x, y, z) satisfying (4) constitute a finite union of convex polyhedra P_1, \dots, P_N in \mathbb{R}^3 . Since (4) is the countably

² Whenever we define a logical theory, we allow (unless explicitly restricted) arbitrary first-order quantification and boolean connectives.

infinite union $\bigvee_{i \in \mathbb{N}} (i \geq 1 \wedge x \leq i \wedge y = i \wedge z = 1)$, it follows that at least one of the convex polyhedra P_j coincides with the union of infinitely many polyhedra of the form (5) ($x \leq i \wedge y = i \wedge z = 1$). This is not possible, because the union of polyhedra of the form (5) is not convex (they are all disjoint). \square

Suppose the user can guess a reachable region like in Definition 3.2 (using the simple heuristic of extrapolating from the first few reachability steps) and that furthermore the guess is *directly inductive* (cf. end of Section 2). Then, verifying that the guess is correct can be done by induction in a completely automatic manner. Indeed, both the base and the inductive steps of the proof require computing the extension and projection operations (cf. equations (1), (2) of Section 2) for formulas of the theory $(\mathbb{R}, \mathbb{N}, +, \leq)$. This amounts to proving finitely many implications of the form $\forall x \in \mathbb{R}^n. \forall i \in \mathbb{N}^m. \exists y \in \mathbb{R}. \varphi(x, i, y) \Rightarrow \psi(x, i, y)$. Proving such an implication can be done automatically, by eliminating the existential quantifiers on the real variables using the Fourier-Motzkin algorithm [Zie95] (transforming the universal quantifiers into existential ones by taking the negation of the formula whenever necessary). At the end we are left with a formula of Presburger arithmetic, which is decidable.

In the situation where the guessed region is not directly inductive, one can still attempt make it directly inductive as indicated in Section 2, by introducing new variables (one of which represents the iteration number) and new constraints connecting the existing and the new variables. Finally, even when a guess is not directly inductive, it can be useful (as an invariant of the system) to prove safety properties. We demonstrate these approaches in Section 4 on some well-known examples.

We now define a class of linear hybrid automata whose reachable region can be defined in terms of natural and real numbers, using addition and multiplication. Consider the theory $(\mathbb{R}, \mathbb{N}, +, \cdot, \times, \leq)$ of reals and naturals with multiplication between naturals \cdot, \times , multiplication between naturals and reals \cdot, \times , and inequality. Any formula in this theory is a boolean combination of *linear* inequalities in the real variables, with *polynomial* coefficients in the natural-number variables; for example, $(n^3 - 1) \cdot x + m \cdot y + n \geq 0$, where x, y are real variables and m, n are natural-number variables.

Definition 3.4 (multiplicative-inductive hybrid automata). A linear hybrid automaton is *multiplicative-inductive* if its reachable region can be expressed as a set of pairs $\{(l, P_l) \mid l \in L\}$ such that for all location $l \in L$, P_l is a formula of the theory $(\mathbb{R}, \mathbb{N}, +, \cdot, \times, \leq)$ with *all the natural-number variables outermost existentially quantified*. \square

The linear hybrid automaton³ in Figure 2 is multiplicative inductive: it can be shown easily that the reachable region at location s_1 is defined by the formula (6) $\exists n \in \mathbb{N}. (n \geq 1 \wedge x = 1 \wedge n \cdot y = 1 \wedge v = 0 \wedge u = 0)$, where x, y, u, v are real variables, and n is a natural-number variable.

³ In Figure 2, activities $\dot{x} = \dot{y} = \dot{u} = \dot{v} = 0$ at all locations are not represented.

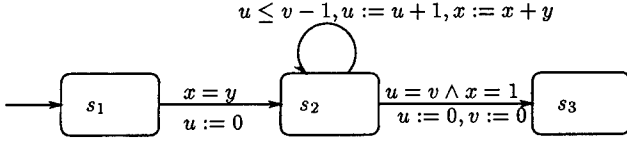


Figure 2. Multiplicative-inductive hybrid automaton

Proposition 3.5. *The class of additive-inductive hybrid automata is strictly included in the class of multiplicative-inductive hybrid automata.*

Proof. The proof of this proposition is based on the following observations. Given two predicates φ and ψ on the real variables x_1, \dots, x_n , we identify φ and ψ with the sets of points in \mathbb{R}^n that they respectively define. We define the *maximal distance* $\Delta(\varphi, \psi)$ between φ and ψ as follows: if φ or ψ are empty then $\Delta(\varphi, \psi)$ is a special value \perp ; otherwise, $\Delta(\varphi, \psi)$ is the lowest upper bound of the set of distances in \mathbb{R}^n between a point satisfying φ and a point satisfying ψ .

Consider now an additive-inductive hybrid automaton, a location l of the automaton, and the formula $\exists i_1 \in \mathbb{N} \dots \exists i_q \in \mathbb{N}. \varphi(x_1, \dots, x_n, i_1 \dots i_q)$ that defines the reachable region of the automaton at location l . Without restricting the generality, it is possible to suppose that formula φ is a convex linear predicate in variables $x_1, \dots, x_n, i_1, \dots, i_q$. We define a sequence $(\varphi_m)_{m \geq 1}$ of linear predicates by the relation $\varphi_m(x_1, \dots, x_n) = \varphi(x_1, \dots, x_n, m \dots, m)$; i.e., the sequence of predicates $(\varphi_m)_{m \geq 1}$ is obtained by replacing in formula φ all integer variables by the value m . Thus, any predicate in the sequence $(\varphi_m)_{m \geq 1}$ is a convex linear predicate on x_1, \dots, x_n ; that is, any predicate φ_m is a convex polyhedron in \mathbb{R}^n .

We now define the sequence $(\Delta_m)_{m \geq 1}$ by $\Delta_m = \Delta(\varphi_m, \varphi_{m+1})$ for all $m \geq 1$. We show that the sequence $(\Delta_m)_{m \geq 1}$ can behave in one of three possible manners. In the first case, there are infinitely many polyhedra φ_m that are empty and thus for infinitely many $m \geq 1$, $\Delta_m = \perp$. Otherwise, there exists an index $M \geq 1$ such that for all $m \geq M$, all polyhedra φ_m are non-empty. Then it can be shown that for all $m \geq M$, each vertex of φ_{m+1} is obtained from some vertex of φ_m by translation by some constant vector $w \in \mathbb{R}^n$. The vector w depends on the vertex but not on the index m . If all such vectors w are 0, then we have the second case: for all $m \geq M$, the polyhedra φ_m are equal, and hence the sequence $(\Delta_m)_{m \geq M}$ is constant. Otherwise, at least one vector w is not 0 and we have the third case: for all $m \geq M$, $\Delta_m \geq |w| > 0$ (where $|w|$ denotes the length of vector w).

Consider now the hybrid automaton in Figure 2 and suppose that it is additive-inductive. We have seen that the formula $(6) \exists i \in \mathbb{N}. (i \geq 1 \wedge x = 1 \wedge i \cdot y = 1 \wedge v = 0 \wedge u = 0)$ represents the reachable region of this hybrid automaton at location s_3 . We apply the previous constructions: we obtain the sequence of predicates $\varphi_m = (x = 1 \wedge m \cdot y = 1 \wedge v = 0 \wedge u = 0)$ and the sequence of distances $\Delta_m = 1/m(m+1)$, for all $m \geq 1$. The last sequence is strictly decreasing and converges to 0. But we have seen that this cannot be the case for a sequence $(\Delta_m)_{m \geq 1}$ obtained (as described above) from the reachable region of an additive-inductive hybrid automaton. Hence, the multiplicative-inductive hybrid automaton in Figure 2 is not additive-inductive. \square

Reachability verification can still be applied to multiplicative-inductive hybrid automata, provided the user guesses the reachable region. For instance, consider the hybrid automaton in Figure 2, whose initial region I is defined by location s_1 . We apply reachability verification: we guess the reachable region at location s_3 to be formula (6) above (using the heuristic of observing the first steps of reachability construction). This guess is furthermore *directly inductive* (cf. end of Section 2): to prove that the guess is correct, we show by induction that for all $k \geq 1$, the region $Post^k(I)$ at location s_3 is described by the formula $(x = 1 \wedge k \cdot y = 1 \wedge v = 0 \wedge u = 0)$. However, unlike the case of additive-inductive hybrid automata, this proof can only be partially automated. Indeed, the extension and projection operations (equations (1), (2) of Section 2) can be computed automatically for predicates in $(\mathbb{R}, \mathbb{N}, +, \cdot, \mathbb{N} \times \mathbb{N}, \cdot, \mathbb{N} \times \mathbb{R} \leq)$: these operations require eliminating the existential quantifiers on the real variables, which can be done using a generalization of the Fourier-Motzkin algorithm [BR97]. But after the quantifier elimination, we are left to decide a first-order formula of the (undecidable) theory $(\mathbb{N}, +, \cdot, \leq)$. This last formula has to be dealt with by theorem proving. So, the verification process is more involved than in the case of additive-inductive hybrid automata.

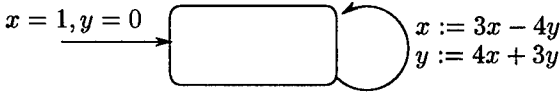


Figure 3. Hybrid automaton with exponential/trigonometric reachable region

While the theory of natural numbers with addition, multiplication and order is extremely expressive for encoding purposes, there exist linear hybrid automata whose reachable regions are most naturally expressed in terms of other operations, like exponentials and trigonometric functions. Consider the hybrid automaton in Figure 3. The transition sets the variables to new values⁴ that we denote x', y' . Let $\theta \in \mathbb{R}$ be such that $5 \cos \theta = 3$. Then, we have $x' = 5(x \cos \theta - y \sin \theta)$, $y' = 5(x \sin \theta + y \cos \theta)$. Interpreted as a vector operation, the previous relations just say that vector $[x', y']$ has a length 5 times greater than vector $[x, y]$, and that $[x', y']$ is rotated by angle θ from $[x, y]$. Thus, the reachable region is defined by formula $\exists n \in \mathbb{N}. \exists \theta \in \mathbb{R}. (x = 5^n \cos n\theta \wedge y = 5^n \sin n\theta \wedge 5 \cos \theta = 3)$. This would suggest that reachable regions need quite expressive theories in order to be expressed finitely. However, it is easy to show that the previous region can be encoded in the first-order theory of integers with multiplication: let $code(x, y)$ be an encoding function of pairs of integers as natural numbers, and consider the natural numbers of the form (7): $2^{code(x_1, y_1)} \cdot 3^{code(x_2, y_2)} \cdot \dots \cdot p_n^{code(x_n, y_n)}$. Here, p_n is the n -th prime number, and x_n, y_n are the terms of the sequence defined by $x_1 = 1, y_1 = 0$, and the transition relation of the automaton. Clearly, the fact that (x_n, y_n) is in the reachable region is encoded by the existence of natural numbers of the form (7), which can be described in the theory $(\mathbb{N}, +, \cdot, \leq)$.

⁴ The linear assignments can be simulated by appropriate slopes, tests, and resets.

Finally, we mention a restricted subclass of linear hybrid automata for which the reachable region can be computed algorithmically, even though reachability construction does not necessarily terminate. Some well-known examples of hybrid automata (like the ones we discuss in Section 4) are in this class. We say a hybrid automaton is *time-predictable* if for each location l' and each pair of transitions (l, l') and (l, l'') with destination (resp. with origin) l , there exists an interval of \mathbb{R}^+ such that transition (s', s'') can be fired at any moment within the given interval, after the firing of transition (s, s') . We say a hybrid automaton is *without nested cycles* if its graph is equivalent to a regular expression (on the transition names) without nested $*$ operations. We have proved⁵ that time-predictable hybrid automata without nested cycles are additive-inductive but not polyhedral (cf. Definitions 3.1, 3.2), and that their reachable region can be computed algorithmically, by a procedure different from reachability construction. This shows that there exist hybrid automata for which the reachability problem is decidable, even though reachability construction does not terminate.

4 Hybrid Automata in PVS

We outline the modeling of hybrid automata and reachability verification in Pvs [ORR⁺96]. First we specify a theory `polyhedra[n]` of n -dimensional polyhedra (parametric in the dimension $n \in \mathbb{N}$). It contains essentially the definitions of extension, projection (formulas (1), (2) of Section 2), and intersection operations on polyhedra. Writing such first-order predicates in the higher-order Pvs specification language is straightforward. Then, we write another theory that is specific to the particular hybrid automaton to be analyzed (containing the definition of the automaton features: states, transitions, activities, invariants, guards, and resets). This second theory uses (imports) the theory `polyhedra[n]`, instantiating n with the number of variables of the hybrid automaton. Finally, in a third theory called `symbolic-analysis` we specify the types and operations of reachability analysis (independent of any particular hybrid automaton): the `region` type (record of state and polyhedron), the continuous and discrete successors of a region, and a `post` predicate on regions, according to the definition of the *Post* operation (cf. Section 2):

```
region : TYPE = [# thestate: state, thepoly: poly #]

continuous(r1:region) : region =
  (#
    thestate:= thestate(r1),
    thepoly:= intersection(extend(thepoly(r1),
                                slope(thestate(r1))), invar(thestate(r1)))
  #)
```

⁵ The proof is not presented here due to lack of space.

```

discrete (r1:region, t:trans) : region =
  (#
    thestate := dest(t),
    thepoly:= intersection(project(reset(t),
                                intersection(thepoly(r1),guard(t))),
                                invar(dest(t)))
  #)

post(R1,R2:setof[region]) : bool =
  FORALL (r2:region): member (r2,R2)
  IMPLIES EXISTS(r1:region,t:trans):
    member(r1,R1) AND orig(t)=thestate(r1)
    AND r2 = continuous(discrete(r1,t))

```

To prove statements about the reachable region $Post^*(I)$, we use induction and the predicate `post`. We now describe the application of reachability verification to examples of hybrid systems modeled by additive-inductive hybrid automata.

The leaking gas burner. The hybrid automaton in Figure 4 models a leaking gas burner [CHR91]: location s_1 (resp. s_2) stands for the leaking (resp. the non-leaking) state of the system; variable x is used to control the time spent in each state, variable y is a global clock, and variable z measures the total time spent by the gas burner in the leaking state. A design requirement for the leaking

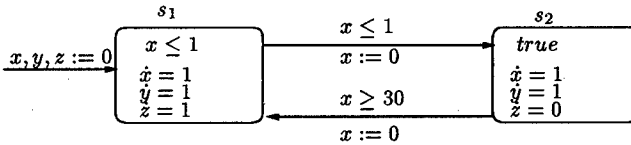


Figure 4. Leaking gas burner automaton

gas burner is that in any interval of time of at least 60 seconds, the leaking time does not exceed 5% of the total time. This can be expressed by the fact that linear predicate $y \geq 60 \Rightarrow 20z \leq y$ is an invariant of the system (i.e., true in all reachable states). The specification in PVS of this example includes the theories `polyhedra[n]` with n instantiated by 3 (the number of variables of the automaton), and `symbolic-analysis` for the reachability analysis of the system. The system itself (hybrid automaton in Figure 4) is specified in a theory `leaking-gas-burner`, that contains the description of the automaton: locations with invariants and differential laws, and transitions called `s1_to_s2` and `s2_to_s1`, with their guards and variables to reset. The reachability construction does not terminate⁶ but by studying the first few iterations, one can guess

⁶ Although *backwards* reachability construction terminates in this case.

that the reachable region is described by the following set of linear regions (from which it can be seen that the hybrid automaton is additive-inductive):

$$\{\langle s_1, 0 \leq x \leq 1 \wedge x = y = z \vee \exists i \in \mathbb{N}. (i \geq 1 \wedge 0 \leq x \leq 1 \wedge 0 \leq z - x \leq i \wedge 30i + z \leq y) \rangle, \langle s_2, 0 \leq z \leq 1 \wedge y = x + z \wedge x \geq 0 \vee \exists i \in \mathbb{N}. (i \geq 1 \wedge 0 \leq x \wedge 0 \leq z \leq i + 1 \wedge 30i + x + z \leq y) \rangle\}$$

However, this guess is not directly inductive (cf. end of Section 2) because the natural-number variable i does not represent the number of iterations. It is possible to make the guess directly inductive, by introducing a new natural-number variable j and two new constraints $j = 2i$, $j = 2i + 1$. More precisely, we define the sets of regions $\mathcal{R}(j)$ such that for all $j \geq 2$, $\mathcal{R}(j)$ is equal to:

$$\{\langle s_1, \exists i \in \mathbb{N}. (i \geq 1 \wedge j = 2i \wedge 0 \leq x \leq 1 \wedge 0 \leq z - x \leq i \wedge 30i + z \leq y) \rangle, \langle s_2, \exists i \in \mathbb{N}. (i \geq 1 \wedge j = 2i + 1 \wedge 0 \leq x \wedge 0 \leq z \leq i + 1 \wedge 30i + x + z \leq y) \rangle\}$$

Furthermore, $\mathcal{R}(0)$ equals $\{\langle s_1, 0 \leq x \leq 1 \wedge x = y = z \rangle, \langle s_2, false \rangle\}$ and $\mathcal{R}(1)$ equals $\{\langle s_1, false \rangle, \langle s_2, 0 \leq z \leq 1 \wedge y = x + z \wedge x \geq 0 \rangle\}$. Now, the new “guess” $\exists j \in \mathbb{N}. \mathcal{R}(j)$ is directly inductive (with j representing the number of iterations). We prove by induction on j that $Post^j(I) = \mathcal{R}(j)$, for all $j \in \mathbb{N}$. This means that $Post^*(I) = \exists j \in \mathbb{N}. \mathcal{R}(j)$; i.e., the guess of the reachable region is correct.

Finally, to prove the design requirement of the gas burner $y \geq 60 \Rightarrow 20z \leq y$, we prove that it is implied by $Post^*(I)$. Except for some details (like the expansions of the definitions for *continuous*, *discrete*, *post* etc), PVS can do all the proofs automatically, using its built-in decision procedures.

The reactor temperature controller. This example is taken from [JLHM91]. It is a variant of the nuclear reactor temperature control problem, in which non-linear evolutions are approximated by piecewise-linear functions [HHWT98]. The reactor automaton (cf. Figure 5) has three locations: in the *no-rod* location,

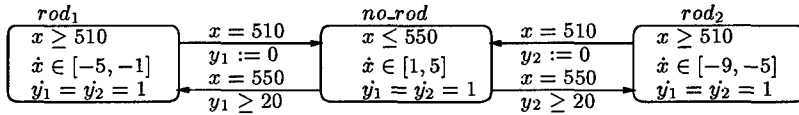


Figure 5. Reactor temperature control automaton

the temperature x increases according to the law $\dot{x} \in [1, 5]$, and control can stay in location *no-rod* as long as the temperature does not exceed 550. When the temperature reaches 550, the reactor uses one of two cooling rods, and the control goes to a location where temperature decreases, according to law $\dot{x} \in [-5, -1]$ or $\dot{x} \in [-9, -5]$, depending on the cooling rod that is used. When the temperature falls to 510, the rod is removed and the reactor goes back to the *no-rod* location. After a rod has been used, it cannot be used again before 20 time units. This is specified using two clocks y_1 and y_2 : when the control leaves the location *rod_i* (that is, rod i is removed from the reactor) the clock variable y_i is reset, and the next entry to location *rod_i* is guarded by the condition $y_i \geq 20$. A design

requirement for the temperature control system is that the temperature never reaches the upper limit ($x = 550$) in the *no_rod* location of the automaton with both rods unavailable ($y_1 < 20$ and $y_2 < 20$). The reachability construction from the initial region (location *no_rod*, variables $x = 510$, $y_1 = y_2 = 20$) does not terminate. However, the reachable region behaves in a regular manner; by studying the output of the model checker HyTech, it can be guessed that the reachable region for location *no_rod* (the location that interests us) has the form:

$$\begin{aligned} & (x \leq 550) \wedge [(y_1 = y_2 \wedge x \geq y_1 + 490 \wedge x \leq 5y_1 + 410) \vee \\ & \exists i \in \mathbb{N}. (x \geq y_1 + 510 \wedge x \leq 5y_1 + 510 \wedge y_2 \geq y_1 + 36 + 28i \wedge y_2 \leq y_1 + 100 + 80i) \vee \\ & \exists i \in \mathbb{N}. (x \geq y_1 + 510 \wedge x \leq 5y_1 + 510 \wedge y_2 \geq y_1 + 16 + 28i \wedge y_2 \leq y_1 + 80(1 + i)) \vee \\ & \exists i \in \mathbb{N}. (x \geq y_2 + 510 \wedge x \leq 5y_2 + 510 \wedge 9y_1 \geq 9y_2 + 112 + 220i \wedge y_1 \leq y_2 + 48(i + 2)) \vee \\ & \exists i \in \mathbb{N}. (x \geq y_2 + 510 \wedge x \leq 5y_2 + 510 \wedge 9y_1 \geq 9y_2 + 292 + 220i \wedge y_1 \leq y_2 + 68 + 48i)]. \end{aligned}$$

We prove in PVS that the above predicate is an invariant at location *no_rod* of the automaton. For this, we show that our guess \mathcal{R} satisfies $I \subseteq \mathcal{R}$ and $\text{Post}(\mathcal{R}) \subseteq \mathcal{R}$. This is enough for proving the design requirement: indeed, the above predicate implies the negation of the ‘dangerous’ region $x = 550 \wedge y_1 < 20 \wedge y_2 < 20$, so the design requirement is met. Except for details like definition expansion, these proofs are completely automatic in PVS.

5 Conclusion

We have presented a new approach to the reachability problem of hybrid automata. The idea is to guess the form of the reachable region and to use theorem proving for verifying that the guess is correct. We have classified hybrid automata according to the theory in which their reachable region can be written finitely. In this classification, we have identified the *additive-inductive* and *multiplicative-inductive* hybrid automata, for which the guess can be done using a simple heuristic and the verification by induction. We have presented some applications using the prover PVS. In the future, we plan to automate the method as much as possible (including automated guess heuristics and adapted strategies for the PVS proofs) for being able to cope with larger examples.

Related work. [BW94] exploit the regularity of cycles on a discrete model (automata with counters). Their approach is fully automatic but it is limited to linear operations on the variables that are idempotent. [BBR97] present a similar approach for a restricted class of hybrid automata (there is a fixed interval of time between transitions), but their method is fully automatic. Abstract interpretation of hybrid automata [HPR94] would automatically recognize the regularities of polyhedra and detect an invariant which, in general, is only an over-approximation of the actually reachable states. Finally, [VH96] describe an approach based on stepwise refinement for the verification of hybrid systems, where PVS is used to prove the correctness of each refinement step.

Acknowledgments. Thanks to Natarajan Shankar, Luca de Alfaro, Peter Habermehl, and the anonymous reviewers of the Hybrid Systems workshop for useful comments and suggestions.

References

- ACH⁺95. R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.
- AD94. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
- AHH96. R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996.
- BBR97. B. Boigelot, L. Bronne, and S. Rassart. An improved reachability analysis method for strongly linear hybrid systems. In *Proc. of the 9th Conference on Computer-Aided Verification, CAV'97*, LNCS 1254, pages 167–178. Springer-Verlag, 1997.
- BR97. A. Burgueño and V. Rusu. Task-system analysis using slope-parametric hybrid automata. In *Proc. of the 3rd Conference on Parallel Processing, Euro-Par'97*, LNCS 1300, pages 1262–1273. Springer-Verlag, 1997.
- BW94. B. Boigelot and P. Wolper. Symbolic verification with periodic sets. In *Proc. of the 6th Conference on Computer-Aided Verification, CAV'94*, LNCS 818, pages 55–67. Springer-Verlag, 1994.
- CHR91. Z. Chaochen, C.A.R. Hoare, and A.P. Ravn. A calculus of durations. *Information Processing Letters*, 40:269–276, 1991.
- HHWT98. T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. Algorithmic analysis of non-linear hybrid systems. *IEEE Transactions on Automatic Control*, 1998. To appear.
- HKPV95. T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Proc. of the 27th Annual ACM Symposium on Theory of Computing, STOC'95*, pages 373–382, 1995.
- HPR94. N. Halbwachs, Y.-E. Proy, and P. Raymond. Verification of linear hybrid systems by means of convex approximations. In *Proc. of the 1st Static Analysis Symposium, SAS'94*, LNCS 864, pages 223–237. Springer-Verlag, 1994.
- JLHM91. M. Jaffe, N. Levenson, M. Heimdahl, and B. Melhart. Software requirements analysis for real-time process-control systems. *IEEE Transactions on Software Engineering*, 17(3):241–258, 1991.
- KPSY93. Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration graphs: a class of decidable hybrid systems. In *Proc. of the 1st Workshop on Theory of Hybrid Systems*, LNCS 736, pages 179–208. Springer-Verlag, 1993.
- ORR⁺96. S. Owre, S. Rajan, J.M. Rushby, N. Shankar, and M.K. Srivas. Pvs: Combining specification, proof checking, and model checking. In *Proc. of the 8th Conference on Computer-Aided Verification, CAV'96*, LNCS 1102, pages 411–414. Springer-Verlag, 1996.
- RR96. O. Roux and V. Rusu. Uniformity for the decidability of hybrid automata. In *Proc. of the 3rd Static Analysis Symposium, SAS'96*, LNCS 1145, pages 301–316. Springer-Verlag, 1996.
- VH96. Jan Vitt and Josef Hooman. Assertion specification and verification using Pvs of the steam boiler control system. In *Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control*, LNCS 1165, pages 453–472. Springer-Verlag, 1996.
- Zie95. G. M. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics*. Springer-Verlag, 1995.

Subanalytic Stratifications and Bisimulations*

Gerardo Lafferriere¹, George J. Pappas², and Shankar Sastry²

¹ Department of Mathematical Sciences,
Portland State University, Portland, OR 97207
e-mail: gerardo@math.pdx.edu

² Department of Electrical Engineering and Computer Sciences,
University of California at Berkeley, Berkeley, CA 94720
e-mail: gpappas,sastry@eecs.berkeley.edu

Abstract. Decidability results for the verification of hybrid systems consist of constructing special finite state quotients called bisimulations whose properties are equivalent to those of the original infinite state system. This approach has had success in the case of timed automata and linear hybrid automata. In this paper, the powerful frameworks of stratification theory and subanalytic sets are presented and used in order to obtain bisimulations of certain analytic vector fields on analytic manifolds.

1 Introduction

Hybrid systems consist of finite state machines interacting with differential equations. Various modeling formalisms, analysis, design and control methodologies, as well as applications, can be found in [2–4, 10, 13]. The theory of formal verification is one of the main approaches for analyzing properties of hybrid systems. The system to be analyzed is first modeled as a hybrid automaton, and the property to be analyzed is expressed using a formula from some temporal logic. Then, model checking or deductive algorithms are used in order to guarantee that the system model indeed satisfies the desired property.

Many verification algorithms are essentially reachability algorithms which check whether the system can reach certain undesirable regions of the state space. Even though for finite state, discrete systems this approach has had success, when dealing with the infinite state space of a hybrid automaton, model checking algorithms are in danger of not terminating. Decidability results for analyzing hybrid systems consider special finite state quotients of the original infinite state hybrid automaton called bisimulations [11]. Bisimulations are special quotient systems in the sense that checking a property on the quotient system is *equivalent* to checking the property on the original system. If an infinite state hybrid automaton has a finite state bisimulation then the analysis and verification procedure is decidable.

* Research supported by the Army Research Office under grants DAAH 04-95-1-0588 and DAAH 04-96-1-0341.

Obtaining bisimulations for purely discrete, finite state automata is clearly decidable since the underlying state space is finite. Correspondingly, the process of constructing bisimulations for hybrid systems may not terminate because of the infinite cardinality of the continuous state space and dynamics. In this paper, we consider the problem of constructing finite state bisimulations for purely continuous systems. More precisely, *given an analytic vector field on an analytic manifold, a set of initial conditions and a set of unsafe states, we would like to construct a finite state transition system such that checking reachability on the finite graph is equivalent to checking reachability of the original continuous system.*

In order to tackle this problem, the powerful frameworks of subanalytic sets and stratification theory [5, 12, 16] are used. Subanalytic sets are an important class of sets having many desirable “finiteness” properties. For example, relatively compact subanalytic sets have finitely many connected components. In addition, subanalytic sets are closed under intersections, unions, complementation as well as forward images under proper maps and inverse images. Stratification theory allows us to deal with many technical issues concerning sets and their boundaries and is crucial in refining partitions. With these tools we present an algorithm for constructing bisimulations of analytic systems as well as a proof that the algorithm terminates in the case of linear vector fields in \mathbb{R}^2 with real or purely imaginary eigenvalues.

The outline of the paper is as follows: In Section 2 we review the notion of bisimulations as well as the algorithm for computing bisimulations for transition systems. Section 3 presents some basic facts about stratification theory and subanalytic sets and in Section 4 we use these facts to construct bisimulations of analytic vector fields. Finally, Section 5 presents interesting issues for further research.

2 Bisimulations

A more detailed exposition of the material described in this section can be found in [11]. A transition system $H = (Q, \Sigma, \rightarrow, Q_O, Q_F)$ consists of a set Q of states, an alphabet Σ of events, a transition relation $\rightarrow \subseteq Q \times \Sigma \times Q$, a set $Q_O \subseteq Q$ of initial states, and a set $Q_F \subseteq Q$ of final states. The transition system is finite if the cardinality of Q is finite and it is infinite otherwise. A region is a subset $R \subseteq Q$. Given $\sigma \in \Sigma$ we define $Pre_\sigma(R)$ as

$$Pre_\sigma(R) = \{q \in Q \mid \exists p \in R \text{ and } (q, \sigma, p) \in \rightarrow\}$$

and $Pre(R)$ as

$$Pre(R) = \bigcup_{\sigma \in \Sigma} Pre_\sigma(R)$$

Let $\sim \subseteq Q \times Q$ be an equivalence relation on the state space and let Q/\sim denote the resulting quotient space. A \sim -block is a union of equivalence classes. For a region R we denote by R/\sim the smallest \sim -block that contains R . Thus,

Q_0/\sim and Q_F/\sim are \sim -blocks containing the initial and final states respectively. The transition relation \rightarrow_\sim on the quotient space is defined as follows: for $Q_1, Q_2 \in Q/\sim$, $(Q_1, \sigma, Q_2) \in \rightarrow_\sim$ iff there exist $q_1 \in Q_1$ and $q_2 \in Q_2$ such that $(q_1, \sigma, q_2) \in \rightarrow$. The quotient transition system is then $H/\sim = (Q/\sim, \Sigma, \rightarrow_\sim, Q_0/\sim, Q_F/\sim)$.

The quotient system H/\sim is a *bisimulation* of H iff Q_F is a \sim -block and for all $\sigma \in \Sigma$ and all \sim -blocks R , the region $Pre_\sigma(R)$ is a \sim -block. A bisimulation is called *finite* if it has a finite number of equivalence classes. Bisimulations are very important because bisimilar transition systems generate the same language. Therefore, checking properties on the bisimilar quotient is equivalent to checking properties of the original transition system. This is very useful in reducing the complexity of various verification algorithms. In addition, if H is infinite and H/\sim is a finite bisimulation, then verification algorithms for infinite systems (for example, hybrid systems) are guaranteed to terminate. A successful application of this approach for timed automata can be found in [1].

Two states $p, q \in Q$ are bisimilar denoted $p \approx q$ iff there exists a bisimulation \sim such that $p \sim q$. It can be shown that if $p \approx q$ then

1. $p \in Q_F$ iff $q \in Q_F$
2. if $(p, \sigma, p') \in \rightarrow$ then there exists q' such that $(q, \sigma, q') \in \rightarrow$ and $p' \approx q'$
3. if $(q, \sigma, q') \in \rightarrow$ then there exists p' such that $(p, \sigma, p') \in \rightarrow$ and $p' \approx q'$

It should be noted that the notion of bisimulation is very similar to the notion of dynamic consistency [7]. Given a transition system H , the following algorithm computes the bisimilarity partition. The algorithm terminates if the bisimilarity quotient is finite.

Algorithm (Bisimilarity for transition systems)

```

Set  $Q/\sim = \{Q_F, Q \setminus Q_F\}$ 
while  $\exists R, R' \in Q/\sim$  and  $\sigma \in \Sigma$  such that  $\emptyset \subset R \cap Pre_\sigma(R') \subset R$ , do
  refine  $Q/\sim = (Q/\sim \setminus \{R\}) \cup \{R \cap Pre_\sigma(R'), R \setminus Pre_\sigma(R')\}$ 
end while
```

Initially the quotient space consists of two equivalence classes, Q_F and $Q \setminus Q_F$ (here \setminus denotes set difference). The algorithm then checks whether there exist \sim -equivalence classes whose preimage under Pre_σ for some σ is neither empty nor a \sim -equivalence class. If there are none then a bisimilarity quotient has been reached. Otherwise there exists $R, R' \in Q/\sim$ such that $R \cap Pre_\sigma(R') \neq \emptyset$ and $R \cap Pre_\sigma(R')$ is a proper subset of R for some $\sigma \in \Sigma$. Then the algorithm refines the partition by splitting R into $R \cap Pre_\sigma(R')$ and $R \setminus Pre_\sigma(R')$. This procedure is repeated either forever or until a bisimilarity quotient is reached.

Inspired by the above bisimulation algorithm, we would like to have an algorithm for obtaining *finite* bisimulations of analytic vector fields. More precisely, the original transition system consists of a (infinite cardinality) real analytic manifold M and the transition relation is generated by the flow of an analytic vector field. A collection of subsets \mathcal{A} of M can be used to describe initial conditions, guards conditions, invariants as well as undesirable regions of the state

space. These sets typically exist within each discrete location of a hybrid system. Given \mathcal{A} we attempt to partition M into a finite bisimilarity quotient M/\sim . If the attempt is successful, then checking reachability of various elements of \mathcal{A} can be directly done on the finite transition system M/\sim . Even though an algorithm computing bisimulations may not, in general, terminate, it may be feasible to guarantee termination for certain classes of vector fields and sets. In order to tackle these very interesting questions, we will use the framework of subanalytic sets and stratification theory.

3 Subanalytic Sets and Stratifications

3.1 Real analytic functions, Manifolds, and Stratifications

In this section we describe some fundamental properties of *subanalytic sets*. We concentrate on properties which are useful for the purpose of constructing a bisimulation for the flow of a real analytic vector field. The most important result here is the *Stratification Theorem* (Theorem 2). For this and other important results on subanalytic sets the main references are [5, 12, 16]. We begin by recalling several standard concepts and facts (see [6] and [9] for more details). In this paper, “manifold” means finite-dimensional, Hausdorff, second countable manifold. We say a manifold is *real analytic* (C^ω) if the transition maps between local charts are analytic functions on their domains (which are open subsets of \mathbb{R}^n). An *embedded submanifold* S of a manifold M is a topological subspace of M together with a differentiable structure such that the inclusion from S into M is a smooth immersion (i.e. has full rank at every point). A vector field X on the real analytic manifold M is *analytic* if its coordinates in any local chart are analytic. If X is an analytic vector field then any integral curve of X is analytic.

We are interested in intersection properties of sets. From this point of view, infinitely differentiable (C^∞ -) functions are not sufficiently “nice”. For example, it is not hard to construct a C^∞ -function whose zero set is a Cantor-like set. (In fact, *any* closed subset of \mathbb{R} is the zero set of some C^∞ -function.) On the other hand, real analytic functions are free from such pathologies. The following classical result illustrates this point.

Theorem 1. *Let I be an open interval and $f: I \rightarrow \mathbb{R}$ be an analytic function. Let $Z = \{x \in I \mid f(x) = 0\}$. Then either $Z = I$ or Z has no accumulation point in I . Equivalently, if f is not identically zero, then every compact subset of I contains at most a finite number of zeros of f .*

Definition 1. *Let M be a real analytic manifold. An analytic (C^ω) stratification of M is a partition \mathcal{S} of M with the following properties:*

1. *each $S \in \mathcal{S}$ is a connected, real analytic, embedded submanifold of M ,*
2. *\mathcal{S} is locally finite (i.e. every compact subset of M intersects at most finitely many sets in \mathcal{S}),*
3. *given two sets $S, P \in \mathcal{S}$, $P \neq S$, such that $S \cap \overline{P} \neq \emptyset$ then $S \subset \overline{P}$ and $\dim S < \dim P$. (We denote by \overline{P} the closure of P .)*

The sets in a stratification are called strata.

3.2 Semianalytic and subanalytic sets

Let M and N be real analytic manifolds and let $C^\omega(M, N)$ denote the set of analytic functions from M into N . (If $f \in C^\omega(M, N)$ we say f is of class C^ω .) Given an analytic manifold U , we denote by $\Sigma(C^\omega(U, \mathbb{R}))$ the Boolean algebra generated by the sets of the form $\{x : f(x) = 0\}$ or $\{x : f(x) > 0\}$, where $f \in C^\omega(U, \mathbb{R})$.

Definition 2. Let M be a real analytic manifold. A subset A of M is semianalytic in M if for every $p \in M$, there is an open neighborhood U of p in M such that $U \cap A \in \Sigma(C^\omega(U, \mathbb{R}))$. If $A \subset M$ is semianalytic in M we write $A \in \text{SMAN}(M)$.

Definition 3. Let M be a real analytic manifold. Define $\text{SBAN}_{rc}(M)$ and $\text{SBAN}(M)$ by

1. $A \in \text{SBAN}_{rc}(M)$ if and only if there is (N, f, A^*) such that N is a real analytic manifold, $f \in C^\omega(N, M)$, $A^* \in \text{SMAN}(N)$, A^* is relatively compact and $A = f(A^*)$;
2. $A \in \text{SBAN}(M)$ if and only if A is a locally finite union of members of $\text{SBAN}_{rc}(M)$.

We say that A is subanalytic in M if $A \in \text{SBAN}(M)$. It is easy to see that $A \in \text{SBAN}_{rc}(M)$ if and only if A is subanalytic in M and relatively compact. The following properties of subanalytic sets are easily derived from the definitions.

1. $\text{SBAN}(M)$ is closed under locally finite unions and intersections.
2. If $A \in \text{SBAN}(M)$ and $f: M \rightarrow N$ is of class C^ω and proper on \bar{A} , then $f(A) \in \text{SBAN}(N)$. (A function f is proper if $f^{-1}(K)$ is compact whenever K is.)
3. If $A \in \text{SBAN}(N)$ and $f: M \rightarrow N$ is of class C^ω , then $f^{-1}(A) \in \text{SBAN}(M)$.

The following two properties require more subtle proofs. They can be derived from the stratification theorem for subanalytic sets.

4. If $A \in \text{SBAN}(M)$ then $M \setminus A \in \text{SBAN}(M)$.
5. A subanalytic set has a locally finite number of connected components, each of which is subanalytic.

Example 1. Points are subanalytic, and so is any locally finite union of points, for example \mathbb{Z}^n as subset of \mathbb{R}^n . The empty set and M are both in $\text{SBAN}(M)$. Let $a, b \in \mathbb{R}$, $a < b$, then $[a, b]$, $[a, b)$, $(a, b]$ and (a, b) are subanalytic in \mathbb{R} . The open ball $B(p, r)$ centered at p of radius r in \mathbb{R}^n is in $\text{SBAN}(\mathbb{R}^n)$.

Example 2. In general, as is clear from the definition, $\text{SMAN}(M)$ is contained in $\text{SBAN}(M)$. In particular, any semialgebraic subset of \mathbb{R}^n is in $\text{SBAN}(\mathbb{R}^n)$.

The following properties clarify further the relation between subanalytic sets and their ambient space. Here we assume that M is a real analytic manifold.

6. Let N be a C^ω , embedded submanifold of M . Then $A \in \text{SBAN}(M) \implies A \cap N \in \text{SBAN}(N)$.
7. Let N be as in (5). Let $A \subset N$ be relatively compact and $\bar{A} \subseteq N$. Then $A \in \text{SBAN}(N) \implies A \in \text{SBAN}(M)$.
8. For every $p \in M$ and every neighborhood W of p , there exists an open neighborhood V_p of p such that: (a) V_p is relatively compact, (b) $\bar{V}_p \subset W$, and $V_p \in \text{SBAN}(M)$.

Remark 1. Let N be a C^ω , embedded submanifold of M . Then if $A \in \text{SBAN}(N)$ and $N \in \text{SBAN}(M)$ it does not follow that $A \in \text{SBAN}(M)$, as the following example shows.

Example 3. Consider the set $S = \{\frac{1}{n} : n \in \mathbb{N}\}$. As a subset of the open interval $(0, \infty)$ the set S is subanalytic since every compact subset of $(0, \infty)$ intersects S in finitely many points. However, as a subset of \mathbb{R} it is not subanalytic. (See Theorem 1.)

Theorem 2 (Stratification Theorem). *Let M be a real analytic manifold and $A \subset \text{SBAN}(M)$, A locally finite. Then there is a C^ω stratification S of M such that:*

1. $S \subset \text{SBAN}(M)$,
2. S is compatible with A . That is, every set in A is a union of strata from S .

Remark 2. It is possible to obtain stratifications in which the strata have additional properties. We mention one here which will be useful in the proof of Theorem 5. A *block* in M is a relatively compact, connected, C^ω , embedded submanifold S of M such that there exists a C^ω surjective diffeomorphism $\phi: C \rightarrow S$ — where C is the open unit cube in \mathbb{R}^k , and $k = \dim S$ — such that the graph of ϕ is subanalytic in $\mathbb{R}^k \times M$. We will assume from now on that the strata in S are blocks.

The following theorem is very useful in proving that certain sets are subanalytic.

Theorem 3. *Consider any formula F of first order predicate calculus with free variables x_1, \dots, x_n in analytic manifolds M_1, \dots, M_n , which is obtained from formulae in some set \mathcal{F} that involve the x_i and other variables y_j ($y_j \in N_j$, N_j an analytic manifold) by means of the logical operations of conjunction, disjunction, negation, universal and existential quantification. Suppose that the quantifications are locally bounded (i.e., that every time a quantifier Qx_i occurs, with $Q = \exists$ or $Q = \forall$, then, if $S_Q(x_i, y)$ is the scope of Qx_i and y are the other variables that are free in S_Q , it follows that for every compact set K of the y domain there is a compact J of the x_i domain such that, for each $\bar{y} \in K$, " $(Qx_i)S_Q(x_i, \bar{y})$ " is satisfied if and only if " $(Qx_i \in J)S_Q(x_i, \bar{y})$ " is satisfied). Then if the formulae in \mathcal{F} define subanalytic sets, so does F .*

The theorem is simply a consequence of the closure properties of the class of subanalytic sets under Boolean operations and taking direct and inverse images (provided that in the case of direct images the map is proper, see [16]). In view of this result one can, in many cases, prove that a set is subanalytic by writing its definition. The following is an example.

Proposition 1. *Let X be an analytic vector field on \mathbb{R}^n . Let $S \subseteq \mathbb{R}^n$ be a C^ω , embedded submanifold, which is also a subanalytic set. Let $\Gamma = \{q \in S: X(q) \in T_q S\}$ (here $T_q S$ is the tangent space to S at q). Then Γ is subanalytic in \mathbb{R}^n .*

Proof. We can write $\Gamma = X^{-1}(T_q S)$ and a tangent vector (q, v) is in $T_q S$ if and only if

$$q \in S \wedge (v = 0 \vee (\forall \varepsilon (0 < \varepsilon < 1) \implies (\exists p (p \in S \wedge p \neq q \wedge (\exists r > 0 \exists s > 0 (r^2 = \|p - q\|^2 \wedge s^2 = \|v\|^2 \wedge \|s(p - q) - rv\|^2 < \varepsilon^2 r^2 s^2 \wedge \|q - p\|^2 < \varepsilon^2)))))))$$

Moreover, as long as q remains in a compact set, the variables p , r , and s can be restricted to lie in a compact set. \square

The following proposition can be proved similarly and will be used in the proof of the Theorem 5.

Proposition 2. *Let $\phi: (0, 1) \rightarrow R$ be a C^ω surjective diffeomorphism with $R \subset \text{SBAN}_{rc}(\mathbb{R}^n)$. Then $\lim_{s \rightarrow 0} \frac{\dot{\phi}(s)}{\|\dot{\phi}(s)\|}$ and $\lim_{s \rightarrow 1} \frac{\dot{\phi}(s)}{\|\dot{\phi}(s)\|}$ exist.*

A deeper and more central result for our analysis is the following. For a proof see [15].

Theorem 4. *Let \mathcal{A} be a locally finite family of nonempty subanalytic subsets of a real analytic manifold M . For each $A \in \mathcal{A}$, let $F(A)$ be a finite set of real analytic vector fields on M . Then there exists a subanalytic stratification \mathcal{S} of M , compatible with \mathcal{A} , and having the property that, whenever $S \in \mathcal{S}$, $S \subset A$, $A \in \mathcal{A}$, $X \in F(A)$, then either (i) X is everywhere tangent to S or (ii) X is nowhere tangent to S .*

We finish this section with a simple proposition which illustrates some of the good intersection properties that analytic curves have with subanalytic sets. The "finiteness" property indicated in the proposition makes it possible to define transitions between strata in a natural way.

Proposition 3. *Let I be an open interval, M a real analytic manifold and $\gamma: I \rightarrow M$ a real analytic function. Let \mathcal{S} be a C^ω stratification of M by subanalytic sets (that is, $S \in \mathcal{S} \implies S \in \text{SBAN}(M)$). If $[a, b] \subset I$ then there exists a finite partition $\{x_1, \dots, x_n\}$ of $[a, b]$ with the property that for each $i = 1, \dots, n - 1$ there exists a stratum $S_i \in \mathcal{S}$ such that $\gamma((x_i, x_{i+1})) \subseteq S_i$.*

Proof. Consider the family $\mathcal{I} = \{\gamma^{-1}(S) \cap [a, b] : S \in \mathcal{S}\}$. Since $\gamma([a, b])$ is compact and \mathcal{S} is locally finite, the family \mathcal{I} is a finite partition of $[a, b]$. By Property 3 of subanalytic sets the sets in \mathcal{I} are subanalytic in I . By Theorem 2, there exists a C^ω stratification \mathcal{J} of $[a, b]$ compatible with \mathcal{I} . Therefore, \mathcal{J} consists of a finite number of points and open intervals. Moreover, for each $J \in \mathcal{J}$ there exists $S \in \mathcal{S}$ such that $\gamma(J) \subseteq S$, as desired. \square

Example 4. The assumption of subanalyticity in the proposition can not be dropped. Consider the stratification of \mathbb{R}^2 by the following five sets:

$$S_1 = \{(0, 0)\}$$

$$S_2 = \left\{ (x, y) : x > 0 \wedge y = x \sin \frac{1}{x} \right\}$$

$$S_3 = \left\{ (x, y) : x < 0 \wedge y = x \sin \frac{1}{x} \right\}$$

$$S_4 = \left\{ (x, y) : x \neq 0 \wedge y > x \sin \frac{1}{x} \right\} \cup \{(0, y) : y > 0\}$$

$$S_5 = \left\{ (x, y) : x \neq 0 \wedge y < x \sin \frac{1}{x} \right\} \cup \{(0, y) : y < 0\}$$

Notice that S_1 , S_2 and S_3 form the graph of the function $f(x) = x \sin \frac{1}{x}$ ($f(0) = 0$), while S_4 and S_5 denote the region above and the below the graph, respectively. Each set is a C^ω , embedded submanifold of \mathbb{R}^2 and they clearly satisfy the condition on the dimension of the strata in the closure of other strata. Finally, consider the constant vector field $X = \frac{\partial}{\partial x}$. Then the integral curve γ of X through $(0, 0)$ is the x -axis (parameterized by x itself). Therefore, the image by γ of any interval containing 0 intersects both S_4 and S_5 an infinite number of times.

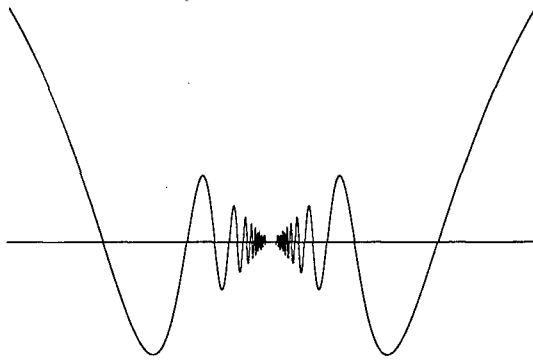


Fig. 1. Infinite crossings on a compact interval

4 Bisimulations of Analytic Vector Fields

Here we describe a process for the construction of a bisimulation for the flow of a real analytic vector field. We assume that we are given a real analytic vector field X on a connected real analytic manifold M as well as a finite family \mathcal{A} of relatively compact subanalytic sets. These sets may describe initial conditions, guards, invariants or undesirable regions of the continuous evolution within a discrete location of a hybrid automaton.

We now invoke Theorem 4 (here there is a single vector field on every stratum) to obtain a stratification \mathcal{S} of M by subanalytic sets which is compatible with \mathcal{A} and such that on each stratum X is either everywhere tangent or nowhere tangent. More precisely, for each $S \in \mathcal{S}$ either: (1) for all q in S , X is tangent to S at q , or (2) for all q in S , X is not tangent to S at q . We now wish to study how the integral curves of X enter and leave each stratum of \mathcal{S} . For this we need a more precise definition of what we mean by *entering* and *leaving* a stratum.

Definition 4. Given two subsets S, T of M , and a real analytic curve $\gamma: I \rightarrow M$ (I an open interval), we say that γ leaves S through T (or enters T from S) if one of the following exiting conditions is satisfied:

- E1** there exist $a, b \in I$ such that $\gamma(t) \in S$ for all $t \in (a, b)$ and $\gamma(b) \in T$
- E2** there exist $a, b \in I$ such that $\gamma(a) \in S$ and $\gamma(t) \in T$ for all $t \in (a, b)$.

The following proposition shows that this definition covers all possible "exiting" situations for strata of \mathcal{S} .

Proposition 4. Let $S \in \mathcal{S}$ and γ be as above. If there exists $t_0, t_1 \in I$ such that $\gamma(t_0) \in S$ and $\gamma(t_1) \notin S$ then there is a stratum T such that either **E1** or **E2** holds.

It is clear from Definition 4 that in case **E1**, $T \cap \bar{S} \neq \emptyset$. By property 3 of a stratification, we conclude $T \subset \bar{S}$ and $\dim T < \dim S$. Similarly in case **E2**, $S \subset \bar{T}$ and $\dim S < \dim T$.

Definition 5. We call a stratum $S \in \mathcal{S}$ tangential if the vector field X is tangent to S at every point of S . We call a stratum transversal otherwise.

The following proposition clarifies further the possible exit situations.

Proposition 5. Let S, T be strata in \mathcal{S} and γ an integral curve of X which leaves S through T . Then one (and only one) of the following holds:

1. condition **E1** holds, S is a tangential stratum and T is a transversal stratum.
2. condition **E2** holds, S is a transversal stratum and T is a tangential stratum.

Our goal is to construct a bisimulation as a quotient of the equivalence relation induced by the stratification \mathcal{S} . More precisely, we would like to define the equivalence relation $\sim_{\mathcal{S}}$ by $p \sim_{\mathcal{S}} q$ iff p, q belong to the same stratum of \mathcal{S} . In $M/\sim_{\mathcal{S}}$ there is a transition from the stratum S to the stratum T iff an integral curve of X leaves S through T . In order to obtain a bisimulation we need the stratification \mathcal{S} to satisfy the following two conditions:

1. if an integral curve of X starting at a point of the stratum S does not exit S , then no other integral curve starting in S leaves S ,
2. whenever an integral curve of X which starts in S leaves the stratum through T , then all other integral curves which start in S leave the stratum through T .

In order to satisfy those conditions we refine the stratification further according to exit features of the integral curves. We describe the iterative process below, which is analogous to the bisimulation algorithm described in Section 2. If the process terminates we obtain the desired bisimulation.

Definition 6 (Refinement Process). *The process has two steps which will need to be iterated. In the first step we refine the tangential strata and in the second we refine the transversal strata.*

Step 1 *Let S be a tangential stratum. For each $T \in \mathcal{S}$, $T \subset \bar{S}$, $T \neq S$ let S_T denote the set of points $q \in S$ for which the integral curve of X through q leaves S through T . Let $S_0 = S \setminus \bigcup S_T$, where the union is taken over all strata T contained in \bar{S} and different from S . So, S_0 is the set of points $q \in S$ such that the integral curve of X through q at time $t = 0$, remains in S for all $t \geq 0$. We subdivide S into the sets S_T and S_0 . This is a finite subdivision of S .*

Step 2 *Let R be a transversal stratum. Let $R^b = \{S \in \mathcal{S} : S \neq R, R \subset \bar{S}\}$. For each $S \in R^b$ and $T \subset \bar{S}$ ($T \neq S$), let R_{S_T} be the set of points $r \in R$ such that the integral curve through r leaves R through S_T . Also, let R_{S_0} denote the set of points $r \in R$ such that the integral curve through r leaves R through S_0 . We subdivide R into the sets R_{S_T}, R_{S_0} where S varies over R^b . This is a finite subdivision of R .*

Remark 3. The new subdivision sets from Step 1 and Step 2 are not in general subanalytic. Therefore, Step 2 requires some clarification since we claim that trajectories "leave R through" one of the sets S_T or S_0 . According to definition 4 we need to verify either **E1** or **E2**. The following proposition gives the key argument.

Proposition 6. *Assume the tangential stratum S is subdivided as in Step 1. Let $\gamma : [t_0, t_1] \rightarrow M$ be an integral curve of X such that $\gamma(t) \in S$ for $t_0 \leq t \leq t_1$. If there exists a set S_* resulting from the subdivision of S such that $\gamma(t_0) \in S_*$ and $\gamma(t_1) \in S_*$ then $\gamma(t) \in S_*$ for $t_0 \leq t \leq t_1$.*

Proof. It follows immediately from the definition of the sets in Step 1, since once a point of a trajectory is in one such set, then for as long as the trajectory remains in S it will belong to the same set. \square

Notation: we will write γ_q to denote the integral curve of X which passes through q at time 0, i.e. with $\gamma_q(0) = q$.

Proposition 7. *With S and R as above, for each $q \in R$ such that γ_q leaves R through S , there exists S_* ($S_* = S_T$ for some T or $S_* = S_0$) such that γ_q leaves R through S_* .*

Proof. Let $\gamma_q : I \rightarrow M$, $a, b \in I$ be such that $\gamma(a) \in R$ and $\gamma(t) \in S$ for $a < t < b$ (we are assuming R is transversal so **E2** holds). Let S_1, \dots, S_k be the sets in the subdivision of S given by Step 1. Let $t_i = \inf\{t \in (a, b) : \gamma_q(t) \in S_i\}$. Then $a = \min\{t_i\} = t_{i_0}$ for some i_0 . We claim that γ_q leaves R through S_{i_0} . To see this let $s_1 \in (a, b)$ be such that $\gamma_q(s_1) \in S_{i_0}$. Suppose there is s with $a < s < s_1$ and $\gamma_q(s) \in S_j$ for $j \neq i_0$. Then there exists s_0 with $a < s_0 < s < s_1$ and $\gamma_q(s_0) \in S_{i_0}$. But this contradicts the previous proposition, so we must have $\gamma_q(s) \in S_{i_0}$ for $a < s \leq s_1$. \square

Notice that in Step 2 we may be subdividing some sets which are in the closure of some tangential set. This requires the iteration of the two steps. In general, we should not expect this process to terminate even if we deal with a finite number of strata or if we limit our study to a compact set. The following example illustrates this point (see Figure 2).

Example 5. Let $M = \mathbb{R}^2$ and X be the linear vector field $\begin{pmatrix} -1 & 1 \\ -1 & -1 \end{pmatrix} \mathbf{x}$. Assume the stratification consists of the following five strata: $S_1 = \{(0, 0)\}$, $S_2 = \{(4, 0)\}$, $S_3 = \{(x, 0) : 0 < x < 4\}$, $S_4 = \{(x, 0) : x > 4\}$, and $S_5 = \mathbb{R}^2 \setminus \bigcup_{i=1}^4 S_i$. The

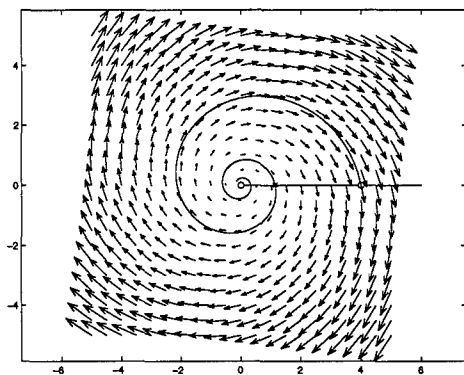


Fig. 2. Process does not terminate

integral curves of X are spirals moving away from the origin. Here S_1 and S_5 are tangential strata. The others are transversal strata. There is no subdivision possible (or necessary) for S_1 . The curves through S_5 exit at one of the three strata S_2 , S_3 , or S_4 . Step 1 requires that we subdivide S_5 into three regions. Two regions are composed of (parts of) the integral curves of X which exit S_5 through S_3 and S_4 respectively. The third is composed of (a part of) the single integral curve which exits through the point S_2 . Step 2 now requires that we subdivide the transversal strata according to a similar rule, but now curves from S_3 leave through three different regions and we must subdivide this stratum

further (into three regions, in fact). The subdivision point corresponds to the first point of intersection of S_3 and the integral curve from S_2 run backwards in time. This now causes one of the regions in S_5 to be subdivided further and clearly the process will not terminate.

For linear vector fields on the plane, the existence of "spiral" points, such as above, is the only obstruction to the procedure as the following theorem illustrates.

Theorem 5. *Let $M = \mathbb{R}^2$, X be the linear vector field Ax and assume that the eigenvalues of A are either real or purely imaginary. Let K be a compact set and define $S_K = \{S \in \mathcal{S} : S \cap K \neq \emptyset\}$ (which is therefore finite). Then the Refinement Process applied to S_K terminates.*

Proof. We will carry out the proof in more generality than necessary. See the remark below.

If $A = 0$ then the process terminates with Step 1, since for each 2-dimensional S , we have $S = S_0$. If $A \neq 0$, then the zero set of A is either $\{(0, 0)\}$ or a line through the origin. We will deal in detail with the first case. The second case can be analyzed with similar methods. We will assume that $\{(0, 0)\}$ is a stratum of \mathcal{S} (\mathcal{S} can be made compatible with $\{(0, 0)\}$). This implies that tangential 1-dimensional strata will not be subdivided further since such a stratum is an arc of a single trajectory of X . Hence, we only need to study the tangential 2-dimensional strata and the transversal 1-dimensional strata (there are no transversal 2-dimensional strata).

The first iteration of the process requires a special analysis. Let $R, S \in \mathcal{S}$, S 2-dimensional and R 1-dimensional, transversal and contained in \bar{S} . Let S_0 be as in Step 1 and R_{S_0} as in Step 2. The following lemma characterizes R_{S_0} .

Lemma 1. *There is a finite stratification of R_{S_0} by subanalytic subsets of \mathbb{R}^2 .*

Proof. Since R is a block (see Remark 2), there is a C^ω diffeomorphism $\phi : (0, 1) \rightarrow R$ whose graph is subanalytic in $\mathbb{R} \times \mathbb{R}^2$. On the interval $(0, 1)$ the connected sets are intervals and hence subanalytic in \mathbb{R} . Therefore, connected sets in R are subanalytic and their boundary (in R) consists of at most two points. To prove the lemma we show that R_{S_0} has finitely many connected components.

Suppose to the contrary that R_{S_0} has infinitely many connected components. Then there exists two infinite sequences $\{s_i\}$, $\{t_i\}$ in the interval $(0, 1)$ such that, for all i , $s_i < t_i < s_{i+1}$, $\phi(s_i) \in R_{S_0}$, and $\phi(t_i) \notin R_{S_0}$. For each i , consider the curve C_i made up of the arcs $\phi([s_i, s_{i+1}])$, $\Gamma_i = \{\gamma_{\phi(s_i)}(t) : t \geq 0\}$, $\Gamma_{i+1} = \{\gamma_{\phi(s_{i+1})}(t) : t \geq 0\}$, and the point $(0, 0)$. This is a continuous, closed, simple curve and therefore it divides the plane into two open connected sets. The trajectory $\gamma_{\phi(t_i)}$ enters one of these sets. This trajectory does not intersect the others and it can not pass through $(0, 0)$. Moreover, it can not leave the region through R since all trajectories cross R in the same direction. Therefore, $\gamma_{\phi(t_i)}(t)$ must remain in the same region for all $t > 0$. On the other hand, by

construction $\gamma_{\phi(t_i)}$ must leave S . Let \bar{t} be the first t such that $\gamma_{\phi(t_i)}(t) \notin S$. So, $q_i = \gamma_{\phi(t_i)}(\bar{t}) \in \bar{S} \setminus S$. The set $S \cup R \cup \{(0, 0)\}$ is connected, relatively compact, subanalytic, and contains each C_i . By construction each q_i is in a different connected component of $\bar{S} \setminus S$, but this is a contradiction since $\bar{S} \setminus S$ has a finite number of connected components. Therefore, so does R_{S_0} as desired. \square

Since the sets S_0 will not be subdivided in subsequent iterations of the refinement process, neither will the sets R_{S_0} . If the stratum $T \in \mathcal{S}$, $T \subset \bar{S}$ is 0-dimensional (i.e. T is a singleton) then R_{S_T} is also 0-dimensional (by uniqueness of solutions). The following lemma characterizes the sets R_{S_T} when T is a 1-dimensional stratum.

Lemma 2. *If $T \in \mathcal{S}$ is 1-dimensional then R_{S_T} is open in R .*

Proof. Let $q \in R_{S_T}$. The trajectory γ_q enters S_T and leaves S through T . Let t_1 be the smallest value of t such that $\gamma_q(t) \in T$. Set $p = \gamma_q(t_1)$. Since X is transversal to R and T we can find relatively open (connected) neighborhoods N_R of q in R , N_T of p in T , and $\delta > 0$, such that $V = \{\gamma_y(t) : y \in N_R, |t| < \delta\}$ and $W = \{\gamma_y(t) : y \in N_T, |t| < \delta\}$ are open, $V \cap W = \emptyset$, $V \cap R = N_R$, and $W \cap T = N_T$. Moreover, we can choose the above so that $\{\gamma_y(t) : y \in N_R, 0 < t < \delta\} \subset S \cap V$ and $\{\gamma_y(t) : y \in N_T, -\delta < t < 0\} \subset S \cap W$. (These constructions are a consequence of basic theorems on differential equations.) We can also assume that the only strata of dimension 0 or 1 which intersect $S \cup V \cup W$ are R and T (because \mathcal{S} is a stratification). Notice that if γ is a trajectory of X such that $\gamma(s) \in W$, then there exists s' such that $\gamma(s') \in W \cap T$ and $\gamma(t) \in W$ for all t between s and s' . By continuous dependence on initial conditions, for all $\varepsilon > 0$ there exist a neighborhood $\tilde{N}_R \subset N_R$ of q in R and $0 < \tilde{\delta} < \delta$ such that $|\gamma_q(t) - \gamma_y(t)| < \varepsilon$, for all $y \in \tilde{V} = \tilde{N}_R \times (-\tilde{\delta}, \tilde{\delta})$ and $0 \leq t \leq t_1$. The set $\Gamma = \{\gamma_q(t) : 0 \leq t \leq t_1\}$ is compact and contained in the open set $S_e = S \cup \tilde{V} \cup W$. Set $d = \frac{1}{3} \text{dist}(q, \{\gamma_q(t) : \delta \leq t \leq t_1\})$. We choose $\varepsilon > 0$ so that $\varepsilon < d$, $\{r : \text{dist}(r, \Gamma) < \varepsilon\} \subset S_e$, and the ball, $B(p, \varepsilon)$, of center p and radius ε , is contained in W . Then for all $y \in \tilde{N}_R$, $\gamma_y(t) \in S_e$ for $0 \leq t \leq t_1$ and $\gamma_y(t_1) \in B(p, \varepsilon)$. We assume further, that $\tilde{N}_R \subset B(q, \varepsilon)$. Let $t_2 = \inf\{t > 0 : \gamma_y(t) \in W\}$ (so $t_2 > 0$ because $V \cap W = \emptyset$). We claim that if $y \in \tilde{N}_R$, then $\gamma_y(t) \in S$ for all $0 \leq t \leq t_2$. Suppose to the contrary that there exists t with $0 < t \leq t_2$ and $\gamma_y(t) \notin S \cup W$. Let $\bar{t} = \inf\{t : 0 < t \leq t_2, \gamma_y(t) \notin S \cup W\}$. Since $\gamma_y(t) \in S$ for $0 < t < \delta$, we know that $\bar{t} \geq \delta > 0$. On the other hand, $\gamma_y(\bar{t}) \in \tilde{V}$. Therefore, $\gamma_y(\bar{t}) \in \bar{S} \cap S_e$ and so $\gamma_y(\bar{t}) \in R \cap \tilde{V} = \tilde{N}_R \subset B(q, \varepsilon)$. But then $\text{dist}(\gamma_q(\bar{t}), q) \leq \text{dist}(\gamma_q(\bar{t}), \gamma_y(\bar{t})) + \text{dist}(\gamma_y(\bar{t}), q) < 2\varepsilon$, which contradicts the choice of ε . Hence, we have $\gamma_y(t) \in S$ for $y \in \tilde{N}_R$ and $0 < t \leq t_2$. By the definition of t_2 and because S is open we can find $\bar{\varepsilon} > 0$ such that $\gamma_y(t) \in S$ for $0 < t < t_2 + \bar{\varepsilon}$ and $\gamma_y(t_2 + \bar{\varepsilon}) \in W$. This implies that γ_y must exit S through T , i.e. $y \in R_{S_T}$. Therefore, $\tilde{N}_R \subset R_{S_T}$ and R_{S_T} is open in R . \square

We continue with the main proof. For a 1-dimensional stratum T we can write $T = \bigcup_{i=1}^{\infty} \phi(J_i)$, with J_i open intervals. For each endpoint a_i of J_i , $x_i = \phi(a_i)$

is in the complement of R_{S_T} . So, either $\{x_i\} = R_{S_{T_i}}$ for some 0-dimensional stratum T_i , or x_i is in the relative boundary of R_{S_0} in R . In either case, there are only finitely many such points x_i . It follows that R_{S_T} has a finite number of connected components.

We have shown that after one iteration we obtain a finite number of singleton sets (points) $\mathcal{P}^{(1)}$ and 1-dimensional connected subanalytic sets $\mathcal{A}^{(1)}$ such that all the new subdivision sets are unions of sets in $\mathcal{P}^{(1)} \cup \mathcal{A}^{(1)}$. Subsequent iterations will only depend on sets of the form R_{S_T} with $T \in \mathcal{P}^{(1)} \cup \mathcal{A}^{(1)}$ (not on R_{S_0}). In fact, if we define for $n \geq 2$, $\mathcal{P}^{(n)}$ as the collection of sets R_{S_T} with $T \in \mathcal{P}^{(n-1)}$, then subsequent subdivisions depend on $\mathcal{P}^{(n)} \neq \emptyset$ (that is, if $\mathcal{P}^{(n)} = \emptyset$ then the Refinement Process terminates). Such sets R_{S_T} are always singletons and there is always a finite number of them. Moreover, by its definition, for each point p with $\{p\} \in \mathcal{P}^{(n)}$ there is a unique $\{q\} \in \mathcal{P}^{(1)}$ and $t_p < 0$ such that $p = \gamma_q(t)$.

Suppose now that the Refinement Process does not terminate. Then there is a trajectory γ of X and an infinite sequence of points x_n with $\{x_n\} \in \mathcal{P}^{(n)}$, and $x_n = \gamma(t_n)$ with $t_n \rightarrow -\infty$. Since we are only subdividing sets in \mathcal{S}_K the sequence must stay in a compact set. We may assume that $\{x_n\}$ converges to a point x_0 . Consider first the case when the eigenvalues of A are real. Then $x_0 = (0, 0)$. Moreover, there exists a one dimensional stratum R of \mathcal{S} which contains infinitely many x_n 's and therefore $(0, 0) \in \bar{R}$. Assume that R is diffeomorphic to the interval $(0, 1)$ via ϕ as above, with $\lim_{s \rightarrow 0} \phi(s) = (0, 0)$. By Proposition 2 there exists $v = \lim_{s \rightarrow 0} \frac{\dot{\phi}(s)}{\|\dot{\phi}(s)\|}$. A direct calculation shows that the following limit also exists: $w = \lim_{t_n \rightarrow -\infty} \frac{\dot{\gamma}(t_n)}{\|\dot{\gamma}(t_n)\|}$. We must have $v \neq -w$, otherwise R can not intersect γ for $|t_n|$ large enough. By changing coordinates and restricting the study to a neighborhood of $(0, 0)$ we may assume that $\{\gamma(t) : t < t_0\}$ and R are both graphs of functions, ψ_γ and ψ_R respectively, with domain $(0, 1)$. At two consecutive intersections s_1, s_2 of these graphs, the vector X must point to opposite sides of the graph of ψ_R . By continuity, for some s , $s_1 < s < s_2$ the vector $X(\psi_R(s))$ must be tangent to the R . This contradicts the transversality of R . Therefore, R and γ can not intersect near $(0, 0)$ and the Refinement Process must terminate. In case A has purely imaginary eigenvalues, $x_0 \neq (0, 0)$ since all trajectories are periodic. Still a similar argument applies using v as above and $w = X(x_0)$. This concludes the proof of the main theorem. \square

Remark 4. The same proof extends to analytic vector fields on the plane with isolated equilibria and with the property that bounded trajectories are either periodic or have "limit" directions (the vector w in the proof). The existence of those limit directions was the only part in the proof that used the linearity of X in an essential way.

5 Conclusions

We presented some preliminary results on obtaining finite bisimulations of analytic vector fields. An algorithm is provided and termination is guaranteed for a class of linear vector fields.

Even though in this paper continuous dynamic systems were considered, the extensions to hybrid systems, even though harder, are conceptually similar. Bisimulations of hybrid systems can still be considered in the framework of subanalytic stratifications by allowing multiple vector fields as well as reset maps. However, the reset maps must be in some sense compatible with the flows for the procedure to terminate. This requirement is already necessary when dealing with a timed automata where the clocks run with irrational slopes.

It should be noted that the main results of this paper are existential since they prove the existence of finite bisimulations. However, there is a long way to making this procedure computationally effective. For certain classes of vector fields the construction can be made effective. For example, one could generalize the decidability result in [8] for multi-polynomial vector fields on the plane, by effectively constructing a finite bisimulation using techniques similar to the ones in this paper (allowing for semialgebraic sets instead of just polyhedra).

Furthermore, if the bisimulation algorithm does not terminate (or is not computable), it may be useful to consider system overapproximations, or abstractions [14], for which the algorithm would terminate (or can be computed).

References

1. R. Alur and D.L. Dill, *A theory of timed automata*, Theoretical Computer Science **126** (1994), 183–235.
2. R. Alur, T.A. Henzinger, and E.D. Sontag (eds.), *Hybrid systems III*, Springer-Verlag, 1996.
3. P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry (eds.), *Hybrid systems II*, Springer-Verlag, 1995.
4. P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry (eds.), *Hybrid systems IV*, Springer-Verlag, 1997.
5. Edward Bierstone and Pierre D. Milman, *Semianalytic and subanalytic sets*, Inst. Hautes Études Sci. Publ. Math. (1988), no. 67, 5–42.
6. William M. Boothby, *An introduction to differentiable manifolds and riemannian geometry*, Academic Press, 1975.
7. P. Caines and Y.J. Wei, *The hierarchical lattices of a finite state machine*, Systems and Control Letters **25** (1995), 257–263.
8. Karlis Cerans and Juris Viksna, *Deciding reachability for planar multi-polynomial systems*, Hybrid Systems III (Berlin, Germany) (R. Alur, T. Henzinger, and E.D. Sontag, eds.), Lecture Notes in Computer Science, vol. 1066, Springer Verlag, Berlin, Germany, 1996, pp. 389–400.
9. J.A. Dieudonné, *Foundations of modern analysis*, Academic Press, 1969.
10. R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel (eds.), *Hybrid systems*, Springer-Verlag, 1993.
11. T.A. Henzinger, *Hybrid automata with finite bisimulations*, ICALP 95: Automata, Languages, and Programming (Z. Fülöp and F. Gécseg, eds.), Springer-Verlag, 1995, pp. 324–335.
12. H. Hironaka, *Subanalytic sets*, In Number Theory, Algebraic Geometry, and Commutative Algebra, in honor of Y. Akizuki, Kinokuniya Publications, 1973, pp. 453–493.
13. O. Maler (ed.), *Hybrid and real-time systems*, Springer-Verlag, 1997.

14. George J. Pappas and Shankar Sastry, *Towards continuous abstractions of dynamical and control systems*, Hybrid Systems IV (Berlin, Germany) (P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, eds.), Lecture Notes in Computer Science, vol. 1273, Springer Verlag, Berlin, Germany, 1997, pp. 329–341.
15. Héctor J. Sussmann, *Subanalytic sets and feedback control*, Journal of Differential Equations **31** (1979), no. 1, 31–52.
16. ———, *Real-analytic desingularization and subanalytic sets: An elementary approach*, Transactions of the American Mathematical Society **317** (1990), no. 2, 417–461.

Integrated Design and Simulation of Hybrid Systems

Georg Lehrenfeld¹, Rolf Naumann², Rainer Rasche², Carsten Rust³ and
Jürgen Tacke³

¹ Heinz Nixdorf Institut, Paderborn University
georg@uni-paderborn.de

² Mechatronics Laboratory, Paderborn University
{naum,rasche}@mlap.uni-paderborn.de

³ C-LAB, Joint R&D Institute of Paderborn University and Siemens Nixdorf
Informationssysteme AG
{car,theo}@c-lab.de

Abstract. In this article we present a new approach for the design of hybrid systems composed of discrete and continuous parts. In our approach the system designers can start their specifications with the discrete as well as with the continuous parts. Both paradigms can be used with their own methodology and Tools. There are integration mechanisms for both paradigms. For the integrated simulation C code is generated. The advantages of our approach are demonstrated by modeling all important aspects of a system for building up motorcades. The model includes a discrete part selecting one of the different strategies modeled in the continuous parts. These are strategies for velocity and distance control for vehicles.

1 Introduction

The increasing complexity of hybrid systems leads to the necessity of improved design methodologies. Hybrid Systems in our context are mechatronic systems with discrete and continuous parts. An important task in hybrid system modeling is to guarantee a correct and well working interaction of all different system parts. This can be achieved by a common design process for the whole system in which every engineer can use his well known design method for discrete or continuous parts. The result is a model of the whole system in which the different parts can be simulated and analyzed together.

There are a lot of tools for the modeling of the different parts. For continuous systems there are Adams, Dymola [EBO96], alaska [Mai93a] and Dads for multi-body systems and for control systems Matlab [Mai93b] and MATRIXx [Int97]. For electrical engineering SABER from Analogly Inc. in Oregon is one of the important tools there. The specification tools for the discrete parts are state oriented. Important tools are STATEMATE [i-L97] for StateCharts, DesignCPN [K.97] for colored petri nets, Skate for Lustre [HCRP91]. Some of these tools, like SABER or Dymola, try to include the whole range of technical disciplines

(mechanics, hydraulics, electronic,...) within the formalism of hybrid systems. In the area of computer science some approaches exist where continuous system parts can be integrated into the discrete modeling languages. In [PL95,WS95] petri net models are extended for the integration of the continuous parts. In [GW96] a graph based formal model is used for hybrid systems. Based on this formalism a tool KANDIS [OGW95] exists for the construction of mixed analog/digital hardware systems. The formal model allows a common graph based modeling of differential equations systems together with discrete time or discrete event systems. Every node in the graph has his own "firing rule". This "firing rule" may either be one for discrete event time systems, one for discrete event systems and one for continuous time systems.

Another very interesting approach for modeling of hybrid systems is the SHIFT [DGS97] language that has been developed within the California PATH project. SHIFT is a programming language for describing dynamic networks of hybrid automata.

In all these approaches the designers of one or even both domains have to specify their parts with tools and in a modeling paradigm not well known. The problem by this approaches is to find an adequate description and representation for every discipline that is practicable for the engineer.

In our approach we describe continuous parts by a system of coupled differential and algebraic equations that can be formulated in three special description languages DSC, O-DSL and O-DSS. O-DSS is used for the topological level, O-DSL for the dynamic nonlinear parts. Descriptions in O-DSS and O-DSL are translated into DSC as base for a process oriented simulator. The discrete parts are described in extended Predicate/Transition-Nets (Pr/T-Nets). They are a form of high-level petri nets with advanced capabilities of transitions and tokens. Transitions can carry first order formulas which have to be calculated by firing transitions. There exist a modeling environment SEA¹ (System Engineering and Animation) developed at C-Lab which offers full support for designing and animation with Pr/T-Nets. Both design methodologies are extended for the integration of models specified in other modeling paradigms. One can start with the continuous parts as well with discrete parts. The other system parts can be integrated in both environments. So, the designer in every field of application can use his own well known modeling paradigm and has to add the other part, designed by another engineer, later. The integrated simulation of the different model parts is realized by generating C code from both modeling environments.

In the following sections we will give a brief introduction in our approaches of modeling discrete and continuous parts with integration techniques of the particular other part. After that we will give a specification example by modeling motorcades in which continuous as well as discrete parts have to be modeled.

¹ prototype available at <http://www.c-lab.de/sea/>

2 Domain specific Methods

2.1 Continuous Modeling

For the last 20 years MLaP has been working on software systems to support the design of mechatronic systems. These systems consist of components from mechanics, hydraulics, electrical engineering, electronics and information processing. In the shape of CAMEL (Computer-Aided Mechatronics Laboratory) [Ric96] a collection of different programs has evolved at MLaP that support the integrative system design from modeling to analysis and synthesis to realization. At the basis of all components is the representation of the system in the computer by means of appropriate description elements. For this purpose three description languages were defined to be employed on different levels of the design process. To describe the system on the topological level there is O-DSS (Objective Dynamic System Structure). The dynamic, nonlinear (and linear) continuous systems are described by a system of coupled differential and algebraic equations that can be formulated in the O-DSL (Objective Dynamic System Language) description language.

```

StateSpaceOdss named: motorType.
  parameter: #(Km) on: ScalarOdss;
             #(Jm) on: ScalarOdss;
  input:    #(alpha) on: ScalarOdss;
             #(getriebeMoment) on: ScalarOdss;
  output:   #(phiMotorP) on: ScalarOdss;
             #(phiMotor) on: ScalarOdss;

  state:    #(phiP) on: ScalarOdss;
             #(phi) on: ScalarOdss;
  auxiliar: #(deltaMoment) on: ScalarOdss;

  auxiliarEquation:
    deltaMoment := ((Km * alpha) - getriebeMoment) * (1 / Jm);

  stateEquation:
    phiP' := deltaMoment;
    phi'  := phiP;

  outputEquation:
    phiMotorP := phiP;
    phiMotor  := phi;
end.

```

Fig. 1. Listing of the motor type

Fig. 1 displays the O-DSL description of the dynamic behavior of a simple engine. The first line defines the system *motorType* from the class of the continuous systems *StateSpaceOdss*. The system interface is defined by the keywords *parameter*, *input*, and *output*. In our case we have the parameters *Km* and

Jm, the inputs *alpha* and *getriebeMoment* and the outputs *phiMotorP* and *phiMotor*. The dynamic equations are given in the body of the system. The behavior of the engine is represented by a linear differential equation of 2nd order that is converted in state-space into a system of two differential equations of 1st order with the following states: the revolution *phiP* and the angle *phi* of the motor shaft following the keyword **state**. After that, we formulate an auxiliary equation for the calculation of the difference torque (*deltaMoment*) between the torque of the motor shaft and that of the gear shaft (**auxiliarEquations**). The differential equations are described in the **stateEquations**. The derivation of a state is expressed by a prime mark (*phiP'*). Eventually the output equations are formulated.

For computer processing the models are represented in the process-oriented description language DSC (Dynamic System Code) [HMN96,Hom97]. The compilation of the model from O-DSS to O-DSL to DSC and the resulting complex transformations (MBS formalisms) are effected by corresponding compilers [HMN96]. For input purposes, a convenient graphical block editor is available that allows formulation and management of the components on the O-DSS and O-DSL levels.

As the systems are becoming ever more complex, it is indispensable to formulate hybrid systems; they are systems that may comprise continuous and discrete system parts. The continuous parts describe the system dynamics while the discrete ones define logical switches that can trigger and manage events. Up to the present, the simulation of complex hybrid systems has been the only possibility of analysis and synthesis; therefore the research on novel methods and procedures is highly topical [Lyg96,Kow97,Eng97]. Connection of the discrete systems to the CAMEL tools available at MLaP can be effected on three different levels:

1. extension of the O-DSL language by elements of discrete components;
2. description of every discrete component in C code blocks with input/output behavior (SEA-Environment) that, along with the continuous parts, can be linked to form a simulator;
3. description of the discrete components by means of a particular tool and coupling on the simulator level.

The degree of the coupling between the discrete and the continuous system parts decreases from top to bottom level. On the first level the system is specified in just one language that gives the engineer easy access to the formulation of hybrid systems; yet, one has to limit work to only particular groups of discrete systems (e.g., automata) in order to guarantee a rather clear language with just a few description elements. Subsequent further processing requires a rather costly extension of the languages and tools underneath by the discrete components (DSC, Simulator). The procedure described on the second level does not necessitate novel description elements; however, it brings about restrictions as to the usability of discrete components. In order to couple the discrete blocks to the continuous ones, the former have to have input/output behavior; they must also be able to read the continuous values and to generate continuous outputs. For

this purpose the interfaces to the components have to be defined unequivocally. The simplest kind of handling hybrid systems is the coupling on the simulator level that will only require synchronization and data exchange (e.g., by means of sockets, DDE, file) of the values (continuous and discrete ones) computed independently of one another.

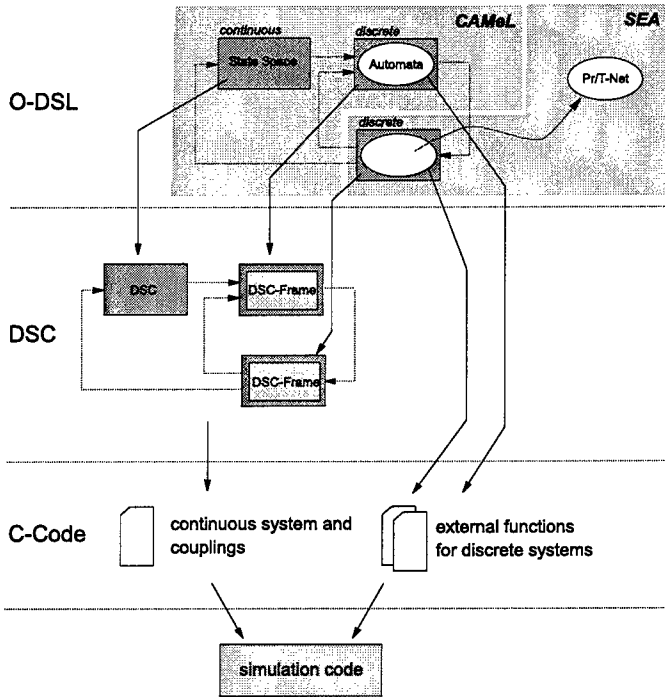


Fig. 2. Integration of discrete system

At the moment MLaP is working on the coupling of discrete systems to continuous ones by way of a combination of levels 1 and 2. For the set of finite automata description elements will be defined (automata, events, and messages) that will serve to generate interface frames on the DSC level that allow coupling to the continuous system.

In order to evaluate the discrete system parts C code will be generated and coupled with the simulation frame (Fig. 2). With this extension it is possible to formulate many kind of hybrid systems (from the view of an engineer), but it is not useful for the description of parallel and non-deterministic systems. A suitable method for these kind of systems are the Pr/T-Nets, that can be described with the SEA-Environment. From SEA a O-DSL frame can be generated that is represented as a block in the CAMEL environment, so the discrete components and the continuous components can be connected graphically. On simulation

level the integration of the Pr/T-Nets is done by linking generated C Code for discrete and continuous components.

2.2 Discrete Modeling

The SEA (System Engineering and Animation)-Environment [KTT97,KKT96] developed at C-LAB offers full support for the modeling and design of the discrete parts of a hybrid system in a modular and distributed manner. It offers a graphical abstraction mechanisms for alternative views of one model and supports the integration of modules specified with other modeling paradigms including continuous ones. The SEA-Environment is based on a formal model with a local paradigm that allows an unambiguous modular specification, namely extended Predicate/Transition-Nets (Pr/T-Nets). They are a form of high level petri nets but they support a more compact specification of complex systems than pure petri nets.

Pr/T-Nets [GL81] are bipartite graphs consisting of places and transitions connected with directed arcs called edges. The places may contain tokens that are consumed and produced by transition. The edges define the "flow" of the tokens. An edge from a place to a transition means the transition consumes tokens from the place and an edge from a transition to a place means that the transition produces tokens on the place. The tokens of a Pr/T-Net are tuples of constants over a set of data types. To further specify the flow in Pr/T-Nets edges may be annotated by sums of constant or variable tuples and transitions may carry first order formulas over a set of constants and variables and a firing rule used to calculate variables occurring on output edges of a transition.

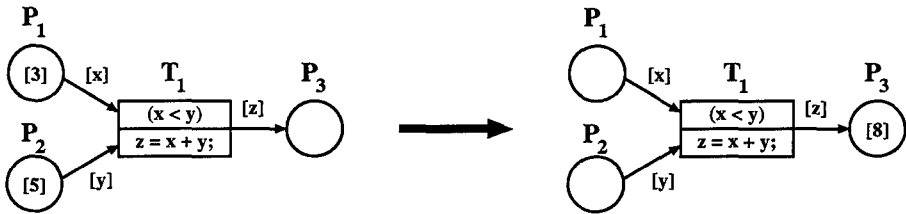


Fig. 3. Pr/T-Net example

Fig. 3 shows a Pr/T-Net consisting of three places (P₁, P₂, P₃) and one transition (T₁). The tokens are integers and the firing rule of T₁ adds the inputs x and y and stores the result in z . The left side of Fig. 3 shows the net before firing of T₁ and the right side afterwards.

We extended the basic definition of Pr/T-Nets in order to support the integration of models specified in other modeling paradigms.

We defined a timing concept to allow the modeling of time dependent system parts. This concept allows the definition of enabling and firing delays for transitions. The enabling delay determines the time delay before a transition may

become active after it has been enabled for any substitution and the firing delay specifies how long a transition is active. If a transition is active, the tokens from the input places are removed but the tokens for the output places are not yet produced.

Furthermore, we allow hierarchical specifications to support a modular specification of complex systems. Places or transitions of an extended Pr/T-Net may be refined by subnets. Such hierarchical nodes have a special semantics which is defined via the activity of their subnet. A subnet of a structured transition is active as long as the structured transition itself is active which is similar to the philosophy of structured nets as described in [CK81]. The subnet of a structured place is active as long as the structured place contains at least one token which is similar to the hierarchical concept in statecharts [Har78].

For an easy integration of textual specification languages we allow code of programming languages within the firing rule of transitions. In our current implementation this can be C/C++ code.

For the integration of graphical specification languages the SEA-Environment allows the definition of an abstract graphical representation for a subnet used to refine a hierarchical node. This representation is also capable to "continuously" represent the system's behavior and state changes during simulation. The description of the abstract representation may use arbitrary graphical elements. Hence, existing graphical specification languages can be (re)produced by using their predefined symbols as the abstract representation of the Pr/T-subnets. All these extensions are explained in more detail in [KKT96].

After the specification of a system part as an extended Pr/T-Net we are able to validate the specification. With the built in Pr/T-Net simulator and -animator the specification can be executed and tested. The readability of the simulation is supported by the animator which shows the abstract graphic representation of the underlying Pr/T-Net models.

If the model satisfies the expectations of the designer C or C++ code can be generated from the model. On the one hand the C++ code can be linked together to conceive a prototype realization of the system. But on the other hand C code including special statements for the integration into the CAMEL tools for the specification of the continuous system parts can be generated.

For the integration of continuous system parts into the SEA-Environment several possibilities exist. One is the integration of C code generated from the CAMEL tools for a continuous model part as annotation of a transition. In this case every time the transition fires one simulation/execution step within the continuous model is performed. The variables at the input edges of the transitions can be used as input for the continuous model and the output can be imported via the variables of the output edges into the Pr/T-Net. Section 3.2 contains an example for this integration method.

A second way is the direct transformation of differential equations into Pr/T-Nets as described in [Bri95]. In this case the differential equations have to be discretized.

In case of the library construction and the direct transformation the continuous models have to be discretized for realizing them as a Pr/T-Net model. This means that the decision for the final implementation technique was already done. With the first integration method several implementation techniques for the continuous model are possible.

After integrating the continuous model parts they must be connected to the discrete parts. Therefore elements from an interface library are used. The right-most element in Fig. 5 is such an element from this library used for converting a discrete value into a continuous one. The token on the place in the middle is copied within every time step to the right place. The value of the token on the right place is changed by adding a token with the new value on the left place.

3 Specification for Motorcades

In this section a part of a mechatronic system serving as an application example for our approach is described. The model is depicted in Fig. 6. It is used for controlling the velocity of a car that shall drive in a motorcade. Inputs for the model are the position of a car driving in front of the considered car ($s_{leading}$), the desired distance between two cars driving in a motorcade (ds_{ref}) and the desired velocity of the car (v_{ref}). Outputs are the velocity (v_{real}) and the position (s_{real}) of the car. The latter is needed by other cars for controlling their velocity.

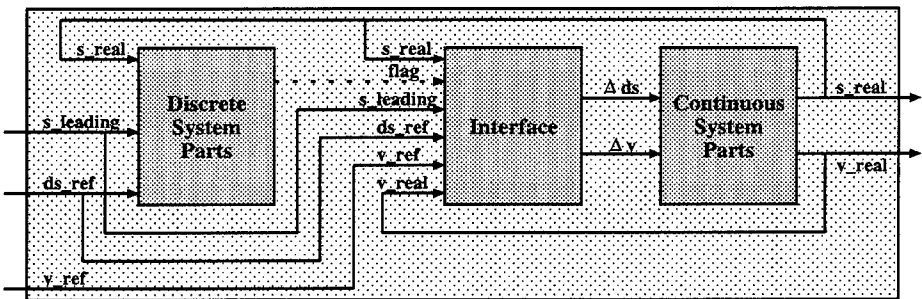


Fig. 6. System with discrete, continuous and interface parts

The whole model mainly consists of three parts. The block 'Continuous System Parts' is needed for computing the position and the velocity of the car, whereas the block 'Discrete System Parts' decides whether the car shall drive with the reference velocity or make up a motorcade with the car driving in front of itself. The continuous block contains units for both controlling the velocity dependent on the reference velocity as well as controlling the velocity dependent on the position of a leading car. At each time one of these models has to be used according to the decision of the discrete system. A third block is needed for interfacing the continuous and discrete model parts.

As can be seen in the figure several interactions between the different system parts are necessary. On the one hand the decision made within the discrete system parts depends on the position of the car computed within the continuous system parts. On the other hand the continuous computations are in turn influenced by the event *flag* triggered asynchronously within the discrete system parts. To summarize all system parts - continuous and discrete ones - have to deal with continuous signals as well as with events triggered asynchronously.

In the following we describe how the example was modeled following our approach to use dedicated tools for modeling the discrete and continuous parts, whereby each tool supports the modeling of interface elements to the particular other part, the generation of code for the other tool and the integration of code generated by the other tool.

3.1 Continuous Parts

With regard to making up motorcades, we consider the longitudinal vehicle behavior. The real vehicle comprises drive and brake trains. The braking pedal acts on the master cylinder transmitting the force to the wheel brake cylinder. This leads to the brakes being actuated. In driving mode, the use of the accelerator alters the throttle valve position. The engine torque depends mainly on the actual engine-rotation speed and the position of the accelerator pedal. This engine torque is being transmitted to the driving wheels via the clutch, the gear-box, the cardan shaft, and the differential. It is transmitted to the tyre and will have an accelerating effect on the car body.

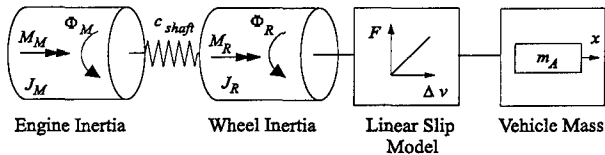


Fig. 7. Linearized model of the drive train

In order to model this complex dynamic behavior of the vehicle, we can make some simplifications and linearize the model [SR96]. An integration of the individual masses allows reduction to three masses for the drive train to be described. One mass represents the engine inertia added to the inertias of the clutch and of the gear (engine inertia). The second mass (wheel inertia) represents the inertias of the gear, shaft, and differential all related to the differential ratio and the inertias of the differential, axis, and tyre. The third mass represents the vehicle body. Fig. 7 displays this simplified substitute model including a linearized tyre model.

In the following, we will deal with the control concepts for longitudinal vehicle dynamics in order to obtain a certain velocity as well as a specified distance

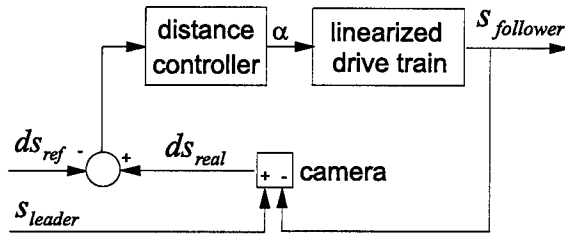


Fig. 8. Structure of distance control

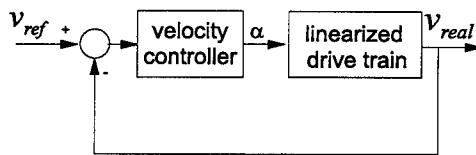


Fig. 9. Structure of velocity control

between two vehicles. Due to the complexity of this hybrid system, we focus on single-input/single-output controllers in order to demonstrate the switch from velocity control to distance control. Fig. 8 and Fig. 9 display the structures of the control strategies. In Fig. 8 the input to the drive train is the throttle angle α , the output is the distance covered by the car body s . The distance covered by the leading vehicle acts as an excitation signal along with the reference distance ds_{ref} . A camera placed in front of the vehicle following up measures the real distance ds_{real} . Fig. 9 sketches the real velocity v_{real} controller with the reference velocity v_{ref} , the controller, the plant and the feedback loop.

The conditions for a switch from velocity control to distance control, when the vehicle detects another one running in front, are specified in the discrete components described in the next section. Since we use linearized drive train models, the switch can be done without triggering a fading process. For implementation, it is important to mention that the inputs and states of the plant such as the throttle angle α , velocities, angles and rotation rates of the drive train model must be "frozen" right before the switch. After the switch, these values serve as the initial conditions for the system with the changed controller structure.

Fig. 10 shows the interaction between the discrete and the continuous components. The interior of the discrete component is provided by the SEA-Environment (cf. Section 3.2). The interface block receives the input values $s_{leading}$, ds_{ref} and v_{ref} and transmits either the distance controller input Δds or the

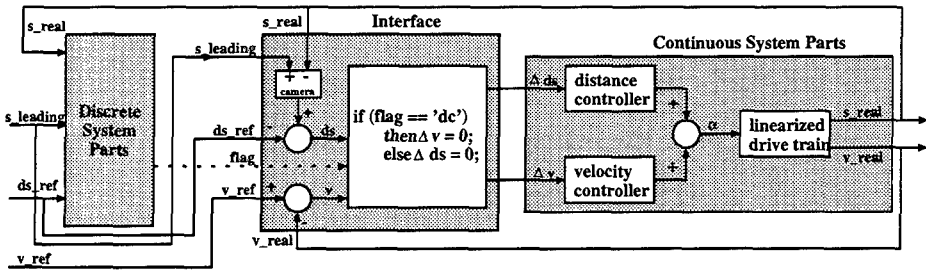


Fig. 10. Block diagram of the hybrid system

velocity controller input Δv in dependency of the *flag* value from the discrete block.

3.2 Discrete Parts

In this section we will describe the Pr/T Net models for the blocks 'Discrete System Parts' and 'Interface' of Fig. 6. Afterwards the integration of C code for the continuous model parts generated by the CAMEL tools (cf. Section 2.1 and 3.1) will be explained.

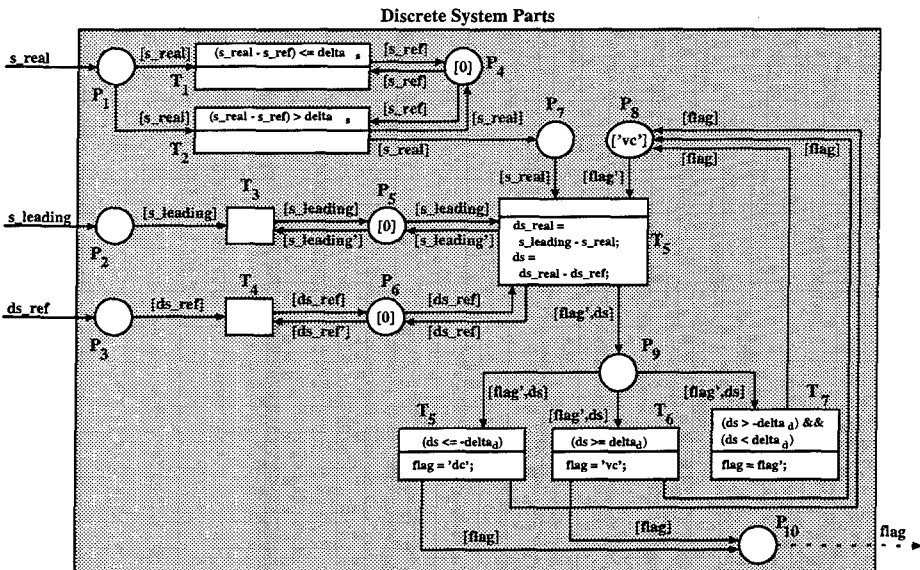


Fig. 11. Pr/T-Net model for the discrete system parts

Fig. 11 shows the Pr/T Net for the block 'Discrete System Parts'. The inputs of this subsystem (s_{real} , $s_{leading}$ and ds_{ref}) are continuous values. The net triggers an event ($flag$) when a switch between velocity control and distance control is necessary within the continuous system parts. A value indicating the mode for computation ('dc' or 'vc') is attached to the event. Checking whether a switch of the mode is necessary takes place whenever the change of the own position (s_{real}) exceeds a certain amount (Δs). This is detected by transition T_2 . If the input value differs less than Δs from the last considered value (s_{ref} , stored in place P_4), it is consumed by transition T_1 . In order to invoke the computation of $flag$ transition T_2 produces a token on place P_7 . The transition T_5 , that becomes enabled due to this token, reads the actual position of a leading vehicle and the actual reference distance from the places P_5 and P_6 . These values are updated by the transitions T_3 and T_4 continuously. Transition T_5 reads the actual mode of computation from the place P_8 and computes the difference ds between the actual distance (ds_{real}) and the reference distance (ds_{ref}). According to these values one of the transitions T_5 , T_6 and T_7 fires changing the mode to a new value if necessary. In order to avoid a permanent switch of the mode a change from velocity control to distance control is not done until the distance of the vehicle falls below a value smaller than the reference distance ($ds_{ref} - \Delta s_d$) and in turn a change from distance control to velocity control is not done until the real distance exceeds ($ds_{ref} + \Delta s_d$).

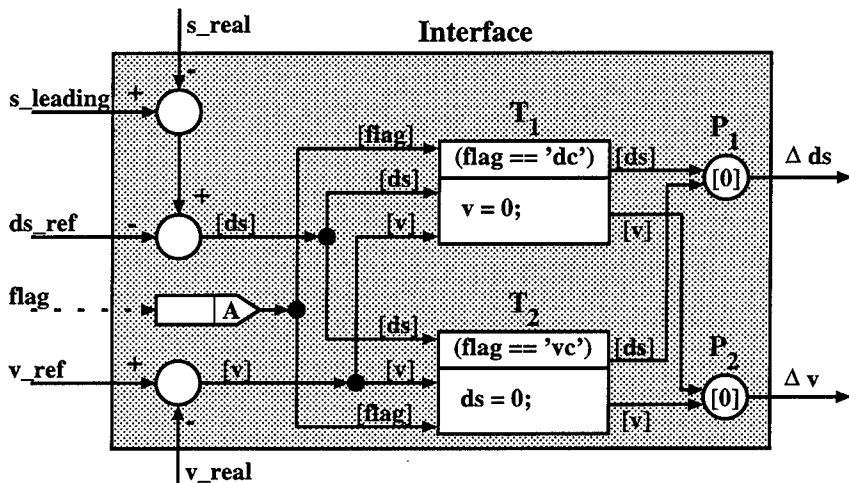


Fig. 12. Pr/T-Net model for the interface parts

The model depicted in Fig. 12 realizes the interface between the discrete system parts and the continuous ones. It is responsible for a continuous production of the values Δds and Δv needed by the continuous system parts. The values depend on the continuous values s_{real} , $s_{leading}$, ds_{ref} and v_{ref} as well as on

the value *flag* which is determined asynchronously by the discrete system parts. The transformation of the event *flag* into a continuous signal is done by an element of the library depicted in Fig. 5. Furthermore the library elements *connect* and *sum* are used for the computation of the output values. The continuous production of the output values is done by the transitions T_1 and T_2 . The value for the controller, that is not active, is set to 0.

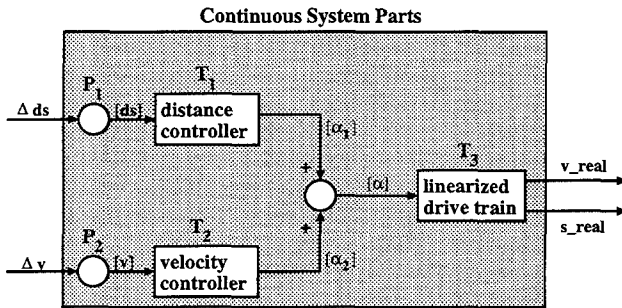


Fig. 13. Pr/T-Net model for the continuous system parts

In order to integrate the continuous model parts into the SEA-Environment the Pr/T-Net depicted in Fig. 13 was specified. The models *distance controller*, *velocity controller* and *linearized drive train* are provided as C code generated by the CAMEL tools. The transitions T_1 , T_2 and T_3 are annotated with functions evaluating the C code for the corresponding continuous models. Since these transitions are enabled within every simulation step the models of the controllers and the drive train are evaluated continuously.

4 Summary

The presented approach of an integrated modeling environment of hybrid systems leaves it up to the designer which part he would like to design first. For the discrete parts he can use the SEA modeling environment for specification with Pr/T-Nets. Special interfaces are available to integrate continuous parts of the system. These parts can be specified with the CAMEL tools. In this environment one can also integrate discrete parts via interfaces to the Pr/T-Net tools. In both environments different possibilities exist for the integration of parts specified in the other modeling paradigm. The one used for the presented example is the integration of C code generated by the other environment. In this case no further investigations are necessary for the integration of the other model parts. So this integration method provides the easiest way for checking the correctness of the discrete parts working together with continuous parts and vice versa. The non trivial example of building up a motorcade shows the usefulness of this approach.

As an advantage of our approach every designer can model in his well known modeling paradigm and can simulate the whole system with all its different parts.

References

- [Bri95] M. Brielmann. Modelling differential equations by basic information technology means. In *Proceedings of the 5th International Conference on Computer Aided Systems, Theory and Technology (EUROCAST'95)*, Innsbruck, Austria, May 1995.
- [CK81] L. A. Cherkasova and V. E. Kotov. Structured nets. In J. Gruska and M. Chytil, editors, *Mathematical Foundations of Computer Science*, volume 118 of *Lecture Notes in Computer Science*. Springer Verlag, 1981.
- [DGS97] A. Deshpande, A. Göllü, and L. Semenzato. The SHIFT Programming Language and Run-time System for Dynamic Networks of Hybrid Automata. Technical report, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 1997. CA 94720.
- [EBO96] H. Elmqvist, D. Bräck, and M. Otter. *Dymola - User's Manual, Version 3.0*. Dynasim AB, 1996.
- [Eng97] S. Engell. Modeling and Analysis of hybrid dynamic systems (in German: Modellierung und Analyse hybrider dynamischer Systeme). *Automatisierungstechnik (at)* 4/97, 1997.
- [GL81] H.J. Genrich and K. Lautenbach. System Modelling with High-Level Petri Nets. *Theoretical Computer Science*, 13, 1981.
- [GW96] C. Grimm and K. Waldschmidt. Kir - a graph-based model for description of mixed analog/digital systems. In *Proceedings of IEEE Euro-DAC/Euro-VHDL*, Genf, Switzerland, September 1996.
- [Har78] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231-274, June 1978.
- [HCRP91] N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The synchronous dataflow programming language LUSTRE. *Proceedings of the IEEE*, 79(9):1305-1320, September 1991.
- [HMN96] M. Hahn and U. Meier-Noe. The Classification Concept in the Object-Oriented Modelling Language Objective-DSS, Exemplified by Vehicle Suspensions. In *Proc. of the IEEE International Symposium on Computer-Aided Control System Design*, Dearborn, Michigan, September 1996.
- [Hom97] C. Homburg. SIMBA - Increasing Efficiency in the Simulation of Heterogeneously modelled Mechatronic Systems. In *Proc. of the 9th European Simulation Symposium (ESS-97) "Simulation in Industry"*, Germany, Passau, October 1997.
- [i-L97] i-Logix Inc. *Statemate MAGNUM Reference Manuals*, 1997.
- [Int97] Integrated Systems Inc. *MATRIXx Reference Manuals*, 1997.
- [K.97] Jensen K. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use. 3, Pratical Use*. EATCS Monographs on Theoretical Computer Science. Springer Verlag, 1997.
- [KKT96] B. Kleinjohann, E. Kleinjohann, and J. Tacke. The SEA Language for System Engineering and Animation. In *Applications and Theory of Petri Nets*, LNCS 1091, pages 307-326. Springer Verlag, 1996.
- [Kow97] J. Kowalewski, S.; Preuaig. Verification of sequential controllers with timing functions for chemical processes. In *Proc. of IFAC 13th World Congress, Vol. J*, pages 419-424, Francisco, USA, 1997.

- [KTT97] B. Kleinjohann, J. Tacke, and C. Tahedl. Towards a Complete Design Method for Embedded Systems Using Predicate/Transition-Nets. In *Proc. of the XIII IFIP WG 10.5 Conference on Computer Hardware Description Languages and Their Applications (CHDL-97)*, pages 4–23, Toledo, Spain, April 1997. Chapman & Hall.
- [Lyg96] J. Lygeros. *Hierarchical, Hybrid Control of Large Scale Systems*. PhD thesis, University of California, Berkeley, 1996.
- [Mai93a] P. et. al. Maier. *alaska 2.0 Handbuch (in German)*. TU Chemnitz, I&M, Chemnitz, 1993.
- [Mai93b] P. et. al. Maier. *MATLAB, High Performance Numeric Computation and Visualization Software*. The Math Works Inc., 1993.
- [OGW95] Peter Oehler, Christoph Grimm, and Klaus Waldschmidt. KANDIS — a tool for construction of mixed analog/digital systems. In *Proc. of IEEE Euro-DAC*, Brighton, September 1995.
- [PL95] S. Pettersson and B. Lennartson. Hybrid Modelling focused on Hybrid Petri Nets. In *Proc. of the 2nd European Workshop on Real-time and Hybrid Systems*, Grenoble, France, June 1995.
- [Ric96] J. Richert. Integration of Mechatronic Design Tools with CAMEL, Exemplified by Vehicle Convoy Control Design. In *Proc. of the IEEE International Symposium on Computer-Aided Control System Design*, Dearborn, Michigan, September 1996.
- [SR96] J. Seuss and J. Richert. Control Structures for Vehicle Convoy Control. In *Proc. of the International Symposium on Advanced Vehicle Control (AVEC-96)*, Monterey, California, June 1996.
- [WS95] R. Wieting and M. Sonnenschein. Extending High-Level Petri Nets for Modeling Hybrid Systems. In *Proc. of the IMACS Symposium on Systems Analysis and Simulation*, Berlin, Germany, June 1995.

Hierarchical Hybrid Systems: Partition Deformations and Applications to the Acrobot System. *

Ekaterina S. Lemch [†] Peter E. Caines ^{‡§}

Abstract

In [3],[5] the notion of dynamical consistency was formulated for hybrid systems so as to define the set of dynamically consistent hybrid partition machines \mathcal{M}^π , $\pi \in \Pi$, associated with a given continuous system \mathcal{S} . This theory includes the notions of a hybrid between-block (HBBC) and in-block controllable (HIBC) partition machine, the lattice $HIBC(\mathcal{S})$ of (HIBC) partition machines, and that of the associated hierarchical-hybrid feedback control systems. In this paper, a brief summary of this theory is presented, and the robustness properties of a partition machine \mathcal{M}^π with respect to deformations of the boundaries of the blocks of π is outlined. An application to the hybrid control of an underactuated double pendulum (Acrobot) system and a fully actuated double pendulum system is then presented. The pendulum example also introduces some properties of the blocks of a state space partition (such as in-block controllability to a distinguished state and controlled umbilical paths) which promise to facilitate the design of hierarchical hybrid control systems. Finally, some global controllability results for nonlinear systems are sketched which have application to the construction of HIBC partitions for mechanical systems.

1 Introduction

In this paper the problem of state quantization, or abstraction, for the design of controllers for continuous systems is treated as a hierarchical control problem using the formulation developed in [6],[4] and [5]). The central notion in this theory is that of dynamical consistency (DC) wherein an abstract transition from one partition block of states to a second is required to satisfy the following condition: every state in the first block can be driven along a trajectory directly

*Work partially supported by NSERC grant number OGP 0001329, NSERC-FCAR-Nortel grant number CRD 180190 and NASA-Ames Research Center grant number NAG-2-1040.

[†]Department of Electrical Engineering, McGill University, 3480 University Street, Montreal, Quebec, Canada H3A 2A7. lemch@cim.mcgill.ca

[‡]Department of Electrical Engineering, McGill University, 3480 University Street, Montreal, Quebec, Canada H3A 2A7. Current address: Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Shatin, N.T.Hong Kong. peterc@cim.mcgill.edu

[§]Also affiliated with the Canadian Institute for Advanced Research.

into the second block without excursions into any other block. This notion permits the definition of the finite state partition machine associated with a given continuous system S and a given (finite analytic) state space partition π (see below and [4],[5]). Consequently, high level hierarchical (or abstracted) control actions maybe be defined and applied to the low level (or base) system. The basic definitions of hierarchical hybrid control theory are reviewed in Section 2 and we note that the recent formulations of finite bisimulations of continuous systems ([1],[7]) involve notions very similar to that of dynamical consistency.

We consider here three aspects of hierarchical hybrid control in the context of general systems and in terms of a controlled double pendulum example.

First, in Section 3, we discuss the general problem of robustness for the high level dynamics of a partition machine \mathcal{M}^π with respect to system partition perturbations. More specifically, we introduce a theory of the robustness of high level (partition machine) dynamics with respect to perturbations of the boundaries of the blocks of a state space partition π . After formally defining a large class of admissible perturbations (or deformations), we give conditions for the high level dynamics of the hierarchical system to be insensitive to such deformations. The condition for such insensitivity to hold for sufficiently small perturbations is that the hybrid in-block controllability (HIBC) condition is preserved for the deformed partition blocks. We note that it is precisely the HIBC condition which is employed in the lattice theoretic formulation of (multilayered) hierarchical control systems in [6],[4] and [5]. Furthermore, the problem of finding conditions for a partition π to be HIBC is equivalent to that of giving conditions for the global controllability of the given system within the subsets of the state space specified by the (open connected) blocks of the partition π .

Second, in Section 4, we treat the construction of partition blocks and DC transitions (or connections) for the underactuated double pendulum system (called the Acrobot) and its fully actuated form (i.e. the double pendulum with torques applied at the shoulder and the elbow). In order to do this we use certain novel state space decompositions, employing blocks possessing the in-block controllability property with respect to distinguished state elements (e.g. equilibrium states) and those which are the tubular neighborhoods of umbilical controlled trajectories. These constructions promise to facilitate the design of hierarchical hybrid control systems. We also construct partition blocks by considering the evolution of certain state subsets under specified controls (in particular friction); this gives rise to boundaries which are (piecewise) invariant manifolds with respect to specified sets of controls. The fully actuated version of the Acrobot is then used to illustrate such constructions of HIBC partitions and, further, their deformation into HIBC partitions. We note that in recent work of Broucke [1] the construction of cell boundaries as invariant sets with respect to specified control laws has also been considered.

Third, in Section 5, we sketch some general controllability results for nonlinear systems and some energy based notions which have application to the construction of HIBC partitions for mechanical systems.

2 Hybrid Partition Machines

Consider the differential system

$$S : \dot{x} = f(x, u), \quad x \in \mathbb{R}^n, u \in \mathbb{R}^m, u(\cdot) \in \mathcal{U}, f : D \times \mathbb{R}^m \subset \mathbb{R}^{n+m} \rightarrow \mathbb{R}^n, \quad (1)$$

where we assume that

- (i) \mathcal{U} is the set of all bounded piecewise $C^q(\mathbb{R}^1)$ ($q \geq 1$) functions of time with limits from the right (i.e. functions which are q times continuously differentiable, except possibly at a finite number of points in any compact interval, and which are bounded and have limits from the right);
- (ii) $F \in C^1(\mathbb{R}^{n+m})$ and for each $u \in \mathcal{U}$, $F(x, u(t))$ satisfies a global Lipschitz condition on D , uniformly in $t \in \mathbb{R}^1$; and
- (iii) D is assumed to be a closed set and to have a non-empty, path-connected interior.

We shall refer to these conditions on F , D , and \mathcal{U} as the *standard (control ODE) conditions*.

Definition 2.1 A *finite analytic partition* of the state space $D \subset \mathbb{R}^n$ of (1) is a finite pairwise disjoint collection of subsets $\pi = \{X_1, X_2, \dots, X_{|\pi|}\}$ such that each X_i is non-empty, open and path-connected, and is such that

$$D = \bigcup_{i=1}^{|\pi|} (X_i \cup \partial X_i),$$

where, further, the boundary ∂X_i of every block X_i is a locally finite union of connected components of $n - p$ dimensional, $p \geq 1$, analytic manifolds (possibly with boundary). $\Pi(D)$ shall denote the set of all finite analytic partitions of D . For partitions $\pi_1, \pi_2 \in \Pi(D)$, π_2 is said to be *weaker* than π_1 , denoted $\pi_1 \preceq \pi_2$, if, for every $X_i \in \pi_1 = \{X_1, \dots, X_n\}$, there exists $Y_j \in \pi_2 = \{Y_1, \dots, Y_m\}$ such that $X_i \subset Y_j$.

□

For the definition of dynamical consistency we need the following notion: a state $y \in \partial X_i \cap \partial X_j$ is said to be a *facial (boundary) state of the pair of blocks* X_i, X_j if y lies in the relative interior of $n - 1$ dimensional connected components of the boundaries ∂X_i and ∂X_j . The *set of facial (boundary) states* is the set of all states which are the facial states of some pair of blocks.

Definition 2.2 For $\pi \in \Pi(D)$, $\langle X_i, X_j \rangle \in \pi \times \pi$ is said to be a *dynamically consistent (DC) pair* (with respect to S) if and only if either $i = j$, or, if $i \neq j$, for all x in X_i , there exists $u_x(\cdot) \in \mathcal{U}$, defined upon $[0, T_x]$, $0 < T_x < \infty$, and there exists a facial boundary state $y \in \partial X_i \cap \partial X_j$, such that:

- (i) $\forall t \in [0, T_x], \phi(t, x, u_x) \in X_i$, and $\lim_{t \rightarrow T_x^-} \phi(t, x, u_x) = y$;

and for the state y in (i) there exists $u_y \in \mathcal{U}$ defined on $[0, T_y), 0 < T_y < \infty$, such that

$$(ii) \forall t \in (0, T_y), \phi(t, y, u_y) \in X_j;$$

where $\phi(\cdot, \cdot, \cdot)$ in (i) and (ii) denotes the transition function of the vector field $f(\cdot, \cdot)$ with respect to the control functions $u_x, u_y \in \mathcal{U}$ and the initial conditions x, y , respectively. □

Definition 2.3 Given $\pi \in \Pi(D)$, the *hybrid DC partition machine*

$$\mathcal{M}^\pi = \langle X^\pi = \{\overline{X}_1, \dots, \overline{X}_p\}, U = \{\overline{U}_i^j; 1 \leq i, j \leq p\}, \Phi^\pi \rangle,$$

based upon the system \mathcal{S} , is the finite state machine defined by $\Phi^\pi(\overline{X}_i, \overline{U}_i^j) = \overline{X}_j$, for all $i, j, 1 \leq i, j \leq |\pi|$, if and only if $\langle X_i, X_j \rangle$ is DC. $\Pi(\mathcal{S})$ shall denote the set of *all* hybrid partition machines of \mathcal{S} . □

Definition 2.4 Hybrid In-block Controllability

A hybrid partition machine \mathcal{M}^π is called *hybrid in-block controllable* (HIBC) if for every $X_i \in \pi$, and for all $x, y \in X_i$, the following holds:

$$\exists u(\cdot) \in \mathcal{U}, \exists T, 0 \leq T < \infty, (\forall t, 0 \leq t \leq T, \phi(t, x, u) \in X_i) \wedge \phi(T, x, u) = y, \quad (2)$$

i.e. each block $X_i \in \pi$ is *controllable for the system \mathcal{S}* . □

The reader is referred to [3],[5] for more information on all notions introduced in this section.

3 Partition Deformations

In this section we establish conditions under which the dynamics of \mathcal{M}^π are robust with respect to sufficiently small perturbations of the partitions defining \mathcal{M}^π . All of the definitions and results here are taken from [9].

Definition 3.1 A *piecewise analytic (p.a.) deformation of the (finite analytic) partition boundary* $\partial\pi = \bigcup_{i=1}^n \partial X_i$ of D is a function $F : \partial\pi \rightarrow D$ with the following properties

- (i) F is continuous;
- (ii) F gives a 1 to 1 correspondence between $\partial\pi$ and $F(\partial\pi)$;

- (iii) F is *piecewise analytic*, in the sense that for every block X_i , $1 \leq i \leq n$, there exists a finite partition $\{D_{s,m}^i\}$ of the boundary ∂X_i such that

$$\partial X_i = \bigcup_{m=1}^{k_i} C_m^i = \bigcup_{m=1}^{k_i} \left(\bigcup_{s=1}^{t_m} D_{s,m}^i \right),$$

and such that the restriction of the function F to $D_{s,m}^i$, $1 \leq i \leq n$, $1 \leq m \leq k_i$, $1 \leq s \leq t_m$, is an analytic function; and

- (iv) $F(\partial D) = \partial D$.

□

Definition 3.2 Let $\pi = \{X_1, X_2, \dots, X_n\}$ be a finite analytic partition of the set D and let F be any function $F: \partial\pi \rightarrow D$ which maps the partition boundary $\partial\pi$ into D . A partition $\pi' = \{X'_1, X'_2, \dots, X'_{|\pi'|}\}$ of the set D is called a *deformation of π induced by F* if and only if $\partial\pi' = F(\partial\pi)$.

□

Theorem 3.1 Let $\pi = \{X_1, X_2, \dots, X_n\}$ be a finite analytic partition of the compact set D and let F be a p.a. deformation of the partition boundary $\partial\pi$. Then,

- (i) there exists a unique finite analytic partition $\pi'_F = \{X'_1, X'_2, \dots, X'_{|\pi'_F|}\}$ of the set D which is a deformation of the partition π induced by F ; and
- (ii) $|\pi'_F| = |\pi| = n$; in other words, the cardinality of a finite analytic partition is invariant under piecewise analytic deformations.

□

Definition 3.3 Let π' be a deformation of a partition π induced by F , where F is a p.a. deformation of the partition boundary. π' is called an ε -deformation of the partition π (denoted π_ε) if

$$\|\pi - \pi'\| \triangleq \max_{z \in \partial\pi} \|z - F(z)\| = \varepsilon.$$

□

Lemma 3.1 Let $\pi = \{X_1, X_2, \dots, X_n\}$ be a partition of a compact set D and let $\pi' = \{X'_1, X'_2, \dots, X'_n\}$ be any deformation of π induced by F , where F is a p.a. deformation of the partition boundary. Then there exists ε^* , $\varepsilon^* > 0$, such that if $\|\pi - \pi'\| \leq \varepsilon^*$, then for any i , $1 \leq i \leq n$, there exists a unique j , $1 \leq j \leq n$, such that

- (i) $X_i \cap X'_j \neq \emptyset$;
- (ii) $F(\partial X_i) = \partial X'_j$.

(We shall call X'_j the image of X_i under F .)

□

Henceforth, we shall say that a deformation π' is a *small deformation* (of π) if $\|\pi - \pi'\| \leq \varepsilon^*$, where ε^* is specified by Lemma 3.1. Further, by Lemma 3.1, for small deformations we can assume without loss of generality that X'_i , $1 \leq i \leq n$, is the image of X_i under F .

Theorem 3.2 Let $\pi = \{X_1, X_2, \dots, X_n\}$ be an HIBC partition of the set D , and let F be any p.a. deformation of the partition boundary of π such that the induced deformed partition $\pi' = \{X'_1, X'_2, \dots, X'_n\}$ is a small deformation and is HIBC. Then there exists ε , $\varepsilon > 0$, such that whenever $\|\pi - \pi'\| \leq \varepsilon$ it is the case that, for all i, j , $1 \leq i, j \leq n$,

$\langle X_i, X_j \rangle$ is dynamically consistent in the partition machine $\mathcal{M}^\pi \Rightarrow$

$\langle X'_i, X'_j \rangle$ is dynamically consistent in the partition machine $\mathcal{M}^{\pi'}$.

□

We also have the following theorem under the same hypotheses as Theorem 3.2.

Theorem 3.3 Let $\pi = \{X_1, X_2, \dots, X_n\}$ be an HIBC partition of the set D , and let F be any p.a. deformation of the partition boundary of π such that the induced deformed partition $\pi' = \{X'_1, X'_2, \dots, X'_n\}$ is a small deformation and is HIBC. Then there exists ε , $\varepsilon > 0$, such that whenever $\|\pi - \pi'\| \leq \varepsilon$ it is the case that, for all i, j , $1 \leq i, j \leq n$, a block trajectory from X_i to X_j exists in the partition machine \mathcal{M}^π if and only if a block trajectory from X'_i to X'_j exists in the partition machine $\mathcal{M}^{\pi'}$.

□

Theorems 3.2 and 3.3 are illustrated in Figure 1 for a two dimensional system \mathcal{M} ; this system appears in [8] as a continuous system which has the partition machine \mathcal{M}^π . (\mathcal{M}^π itself illustrates the general result of [8] that any finite state machine may be obtained as the partition machine of some continuous base system.) Following the general construction specified in [8], a continuous control system with the vector field displayed in Figure 1 is

$$\dot{z} = r(z)^2 G(z, u) + H(z)w,$$

where:

$$z = (x, y)^T; u = (u_1, u_2)^T \in \mathcal{U}^2; w = (w_1, w_2)^T \in \mathcal{W}^2.$$

$$e_1 = (1, 0)^T; e_2 = (-\frac{1}{2}; \sqrt{3}/2)^T \quad e_3 = (-\frac{1}{2}; -\sqrt{3}/2)^T.$$

$$r(z) = \sqrt{x^2 + y^2}.$$

$$H(z) = \sum_{i=1}^3 \prod_{\substack{j=1 \\ j \neq i}}^3 a_j^2(z), \text{ where, for any } z \in X_i, i = 1, 2, 3, a_i(z) = 0 \text{ and } a_j(z),$$

$j = 1, 2, 3, j \neq i$, is the projection of z on e_j in the direction parallel to e_k , $k = 1, 2, 3, k \neq i, j$. In other words, $a_{j_1}, a_{j_2}, j_1, j_2 \in \{1, 2, 3\}, j_1 \neq i, j_2 \neq i$, are the coordinates of the point z in the basis $\langle e_{j_1}, e_{j_2} \rangle$.

$G(z, u)$ is a continuous extension on the whole space $D = \mathbb{R}^2$ of the function $g(z, u)$ defined on $\partial\pi$ in the following way

$$g(z, u) = \begin{cases} -u_1^2 e_3 - u_2^2 e_2, & \text{if } z \in e_1; \\ -u_2^2 e_1, & \text{if } z \in e_2; \\ -u_1^2 e_2 - u_2^2 e_1, & \text{if } z \in e_3. \end{cases}$$

Observe that the function $H(z)$ vanishes on the boundary of any given cell ∂X_i , $i = 1, 2, 3$, leaving the function $G(z)$ to determine the controlled vector field there. Further, for any interior point $z \in \mathbb{R}^2 - \partial\pi$, a control vector w can be chosen in such a way that it compensates the effect of the the first term in the differential equation and hence guarantees the local controllability at the point z .

For example, in \overline{X}_3 the system \mathcal{M} is defined by

$$\begin{aligned} \dot{z} = (x^2 + y^2) & \left[\left(1 - \frac{3}{2\pi}\varphi\right)(-u_1^2 e_3 - u_2^2 e_2) - \frac{3}{2\pi}\varphi u_2^2 e_1 \right] + \\ & \left(x + \frac{1}{\sqrt{3}}y\right)^2 \left(\frac{2}{\sqrt{3}}y\right)^2 w, \end{aligned}$$

where $\varphi, \varphi \in [0; \frac{2\pi}{3}]$, is the angle between the Ox axis and Oz .

We observe that Figure 1 shows the converse to Theorem 3.2 to be false.

4 Hybrid Control of a Double Pendulum

4.1 Acrobot System

The Acrobot system (see [10],[11]) consists of a two-link rigid pendulum system with one actuator at the the second joint (elbow) and a pinned first joint (shoulder) (see Figure 2); it is termed an *underactuated system* since it is a mechanical system with fewer actuators than degrees of freedom. The particular *swing up problem* addressed in [10], [11] is to change the position of the Acrobot from the downward (down-down) position to the inverted (up-up) position and balance it there.

The equations of motion of the system are ([11]):

$$\begin{aligned} d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + h_1 + \phi_1 &= 0 \\ d_{21}\ddot{q}_1 + d_{22}\ddot{q}_2 + h_2 + \phi_2 &= \tau, \end{aligned}$$

where

$$\begin{aligned} d_{11} &= m_1 l_{c1}^2 + m_2 (l_1^2 + l_{c2}^2 + 2l_1 l_{c2} \cos(q_2)) + I_1 + I_2 \\ d_{12} &= m_2 (l_{c2}^2 + l_1 l_{c2} \cos(q_2)) + I_2 \end{aligned}$$

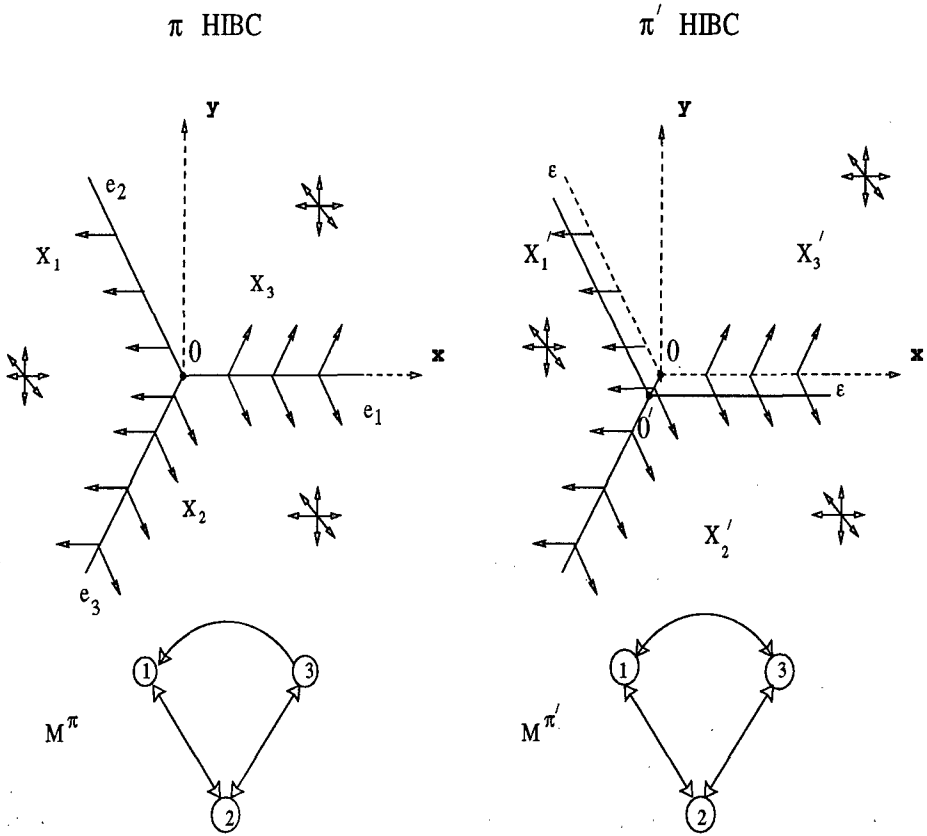


Fig. 1: Illustration of Theorems 3.2 and 3.3

$$d_{21} = d_{12}$$

$$d_{22} = m_2 l_{c2}^2 + I_2$$

$$h_1 = -m_2 l_1 l_{c2} \sin(q_2) \dot{q}_2^2 - 2m_2 l_1 l_{c2} \sin(q_2) \dot{q}_2 \dot{q}_1$$

$$h_2 = m_2 l_1 l_{c2} \sin(q_2) \dot{q}_1^2$$

$$\phi_1 = (m_1 l_{c1} + m_2 l_1) g \cos(q_1) + m_2 l_{c2} g \cos(q_1 + q_2)$$

$$\phi_2 = m_2 l_{c2} g \cos(q_1 + q_2).$$

The control laws for the Acrobot in [11], and the references therein, may be analyzed in terms of the hierarchical hybrid control theory of this paper. This is because the design philosophy of [11] may be described, in general terms, to be to steer the state of the system from one set of states (for instance, a set in a neighbourhood of a particular equilibrium state) to a state in a neighbourhood of another specified equilibrium state. This is to be carried out by invoking a control law L_1 in a given feedback class which is switched to a member L_2 of some other class when the system state enters a specified set.

It has been shown ([11]) that, by using (exact) partial feedback linearization, the first joint (which is not directly actuated) can be driven by the coupling forces arising from the motion of the second joint to trace any given reference trajectory q_1^d . Moreover, subject to such a control, the surface $\{q_1 = q_1^d; \dot{q}_1 = 0\}$ may be shown by use of the Centre Manifold Theorem to define an invariant manifold in the state space which is globally attractive.

The central joint torque τ is chosen so that angle q_1 converges exponentially to a value corresponding to the upright position, while the second pendulum arm (and its corresponding angle) performs a periodic oscillation with respect to the first until the state enters a neighbourhood for the saddle point equilibrium $\{q_1 = \pi/2; \dot{q}_1 = 0\}$. Subject to certain bounds on the initial energy of the system, this set of states is a (controlled) basin of attraction to the up-up position. So when the system when the system enters this set the overall control law may be changed from L_1 to L_2 which consequently stabilizes the Acrobot in the up-up position.

The state space for this problem is

$$D = [0; 2\pi) \times [0; 2\pi) \times \mathcal{R} \times \mathcal{R}.$$

Let $\pi = \{X_1, X_2, \dots, X_n\}$ be a partition (not necessary *HIBC*) of the set D . We shall call any block X_i , $1 \leq i \leq m \leq n$ which is a neighbourhood of a specified state x_i an *in block controllable to x_i* (or $IBC(x_i)$) block whenever any state in

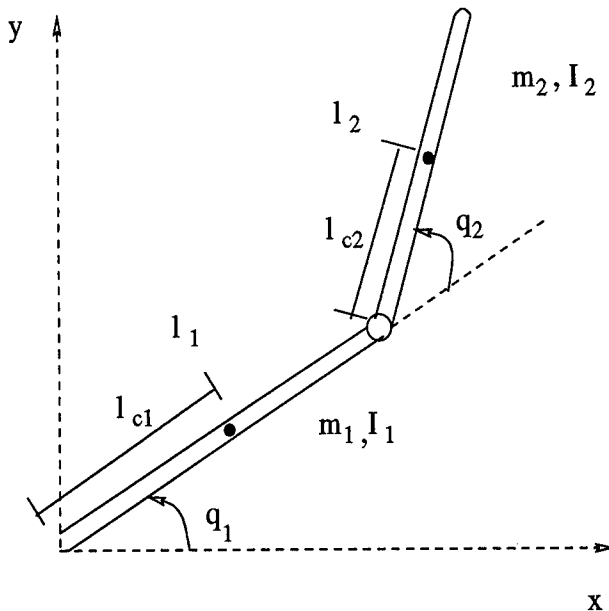
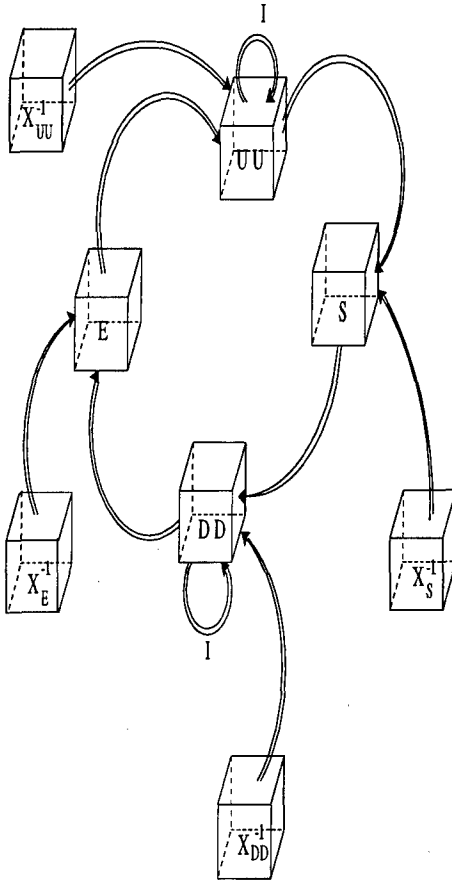


Fig. 2: Acrobot

X_i can be steered (in finite time) into any given neighbourhood of x_i along a trajectory which does not leave the block.

Let X_i be an $IBC(x_e)$ block containing an arbitrary equilibrium state x_e ; such blocks may be shown to exist at any of the equilibrium points of the Acrobot system since it may be shown by a linearization analysis that we may apply an LQR controller at such a point for all sufficiently small angular velocities (whose magnitudes must be used to define each X_i). The $IBC(x_e)$ property of these blocks is indicated in Figure 3 by the self loop bearing the index I.



DD is in-block controllable to the equilibrium point

$$x_e^{dd} = (-\pi/2; 0; 0; 0)$$

UU is in-block controllable to the equilibrium point

$$x_e^{uu} = (\pi/2; 0; 0; 0)$$

E is a tubular neighborhood of paths $DD \rightarrow x_e^{uu}$

S is a tubular neighborhood of paths $UU \rightarrow x_e^{dd}$

X_{UU}^{-1} is the friction-control-inverse (fci) of UU

X_{DD}^{-1} is the fci of the block DD

X_E^{-1} is the fci of the block E

X_S^{-1} is the fci of the block S

Fig. 3: A Partition Machine for the Acrobot System

Consider the two equilibrium points

$$x_e^{uu} = (\pi/2; 0; 0; 0) \text{ and } x_e^{dd} = (-\pi/2; 0; 0; 0),$$

which correspond to the unstable inverted and stable downward positions respectively. Let UU and DD denote $IBC(x_e^{uu})$ and $IBC(x_e^{dd})$ blocks, respectively. It is then possible to specify two f.a. blocks of transition states, E and S , in such a way that

$$\langle UU, S \rangle, \langle S, DD \rangle, \langle DD, E \rangle, \langle E, UU \rangle \text{ are } DC \text{ pairs.}$$

The blocks E and S in the complement of \overline{UU} and \overline{DD} are defined to be tubular neighborhoods (consisting of piece-wise analytic paths) surrounding two selected paths, which we shall call *controlled umbilical paths*. These umbilical paths go from the neighborhoods DD and UU to the points x_e^{uu} and x_e^{dd} , respectively, and hence to the blocks UU and DD , respectively, under the partial feedback linearization control law. Necessarily, the tubular boundaries of E and S are control invariant sets under the piece-wise analytic controls which define them. Hence E and S are indeed f.a. partition blocks, which we note are not in general HIBC.

Now it may be shown that by applying a friction $u^{fri} = -\gamma \dot{q}_2$, $\gamma > 0$, an open set of states \tilde{D} can be driven into the open neighborhood DD of the stable down-down position. Using this observation we can partition a part of the space D

$$D - (E \cup (UU) \cup (DD) \cup S)$$

in the following way.

Define the set X_{UU}^{-1} to be the inverse image of the block UU under the control u^{fri} , i.e.

$$X_{UU}^{-1} = \{x; \exists T \varphi(x, T, u^{fri}) \in \overline{UU} \text{ and}$$

$$\forall t (0 \leq t < T) \varphi(x, t, u^{fri}) \in (\overline{UU} \cup \overline{DD} \cup \overline{E} \cup \overline{S})^c \cap \tilde{D}\}.$$

We can assume without loss of generality that X_{UU}^{-1} is connected. (Otherwise we shrink the UU set until all connected components of the inverse image are joined by trajectories in the open set difference between the old and the closure of the new UU sets.) Hence X_{UU}^{-1} satisfies the hypotheses for a f.a. partition block.

Let $X_{DD}^{-1}, X_E^{-1}, X_S^{-1}$ be analogously defined as the f.a. inverse images of the blocks DD, E, S , respectively. Then, by construction, DC connections can be established for all the following pairs:

$$\langle X_{UU}^{-1}, UU \rangle, \langle X_{DD}^{-1}, DD \rangle, \langle X_E^{-1}, E \rangle, \langle X_S^{-1}, S \rangle.$$

Note, that by the uniqueness of the trajectory for a given control function, the inverse images of the blocks UU, DD, E, S are disjoint. Further, the closure of the union of all the disjoint sets defined here constitute a closed subset D^* of the entire state space D . Thus

$$\pi = \{UU, DD, E, S, X_{UU}^{-1}, X_{DD}^{-1}, X_E^{-1}, X_S^{-1}\}$$

is a partition of the set D^* .

As a result of the analysis above, the partition machine \mathcal{M}^π of the Acrobot is (partly) represented in the Figure 3. This figure gives a high level description of a set of (hierarchical hybrid) controlled behaviours of the Acrobot.

4.2 Fully Actuated Robot

A closely related application to the analysis above of the Acrobot is that of the fully actuated Acrobot, i.e. the fully actuated double pendulum system. The only difference between this system and the Acrobot system is clearly the presence of an input torque to the first equation

$$d_{11}\ddot{q}_1 + d_{12}\ddot{q}_2 + h_1 + \phi_1 = \tau_1$$

$$d_{21}\ddot{q}_1 + d_{22}\ddot{q}_2 + h_2 + \phi_2 = \tau_2.$$

where d_{ij} , h_i , ϕ_i , $i, j = 1, 2$, are defined in the same way as in the underactuated case. The difference between this system and the Acrobot system is the presence of an input torque to the first equation.

Assume that the term d_{12} is nonzero for all values of q_2 . Under this condition, which is called *strong inertial coupling* in [11],

$$\ddot{q}_2 = -\frac{1}{d_{12}}(d_{11}\ddot{q}_1 + h_1 + \phi_1 - \tau_1)$$

and

$$\bar{d}_1\ddot{q}_1 + \bar{h}_1 + \bar{\phi}_1 = \tau_2 - \tau_1 d_{22}/d_{12},$$

where

$$\bar{d}_1 = d_{21} - d_{22}d_{11}/d_{12},$$

$$\bar{h}_1 = h_2 - d_{22}h_1/d_{12},$$

$$\bar{\phi}_1 = \phi_2 - d_{22}\phi_1/d_{12}.$$

A feedback linearizing controller can therefore be defined according to

$$\tau_1 = h_1 + \phi_1 + d_{11}v_1 + d_{12}v_2,$$

$$\tau_2 = \bar{d}_1v_1 + \bar{h}_1 + \bar{\phi}_1 + \tau_1 d_{22}/d_{12},$$

where v_1 , v_2 are additional outer loop control inputs. Using this forced linearization, the system can be represented by the four differential equations

$$\dot{x}_1 = x_2 \quad \dot{x}_2 = v_1,$$

$$\dot{x}_3 = x_4 \quad \dot{x}_4 = v_2,$$

where $x_1 = q_1$, $x_2 = \dot{q}_1$, $x_3 = q_2$, $x_4 = \dot{q}_2$.

For this problem the state space D will be taken to be a specified compact subset of \mathbb{R}^4 , where the states are identified (in the projection into $[-\pi; \pi) \times \mathbb{R} \times [-\pi; \pi) \times \mathbb{R}$) whenever they have identical coordinates except for differences of multiples of 2π in either the first or the third or both of these coordinates.

Let $R_i^I, R_j^{II}, 0 \leq i \leq n, 0 \leq j \leq m$, for some strictly positive integers n and m , be any two sequences of real values such that

- (i) $R_0^I = 0 = R_0^{II}; \quad R_n^I = \pi = R_m^{II};$
- (ii) $R_{i-1}^I < R_i^I, \quad \text{for any } i, 1 \leq i \leq n;$
- (iii) $R_{j-1}^{II} < R_j^{II}, \quad \text{for any } j, 1 \leq j \leq m.$

Consider a collection of sets

$$\pi = \{X_{ij}; 0 \leq i \leq n, 0 \leq j \leq m\}, \quad \|\pi\| = nm,$$

where each X_{ij} is defined in the following way

$$X_{ij} = \{(x_1, x_2, x_3, x_4); R_{i-1}^I < x_1^2 + x_2^2 < R_i^I; R_{j-1}^{II} < x_3^2 + x_4^2 < R_j^{II}\}.$$

Each set X_{ij} is non-empty, path-connected (since any two points $a, b \in X_{ij}$ lie on the surface of a torus which is a path-connected subset of X_{ij}) and open. Further, the boundary of each such set is a finite union of connected analytic manifolds. Any two sets of the collection π are disjoint by the definition. The union of all \bar{X}_{ij} constitute the whole state space D . Hence, by Definition 2.1, π is a finite analytic partition of D .

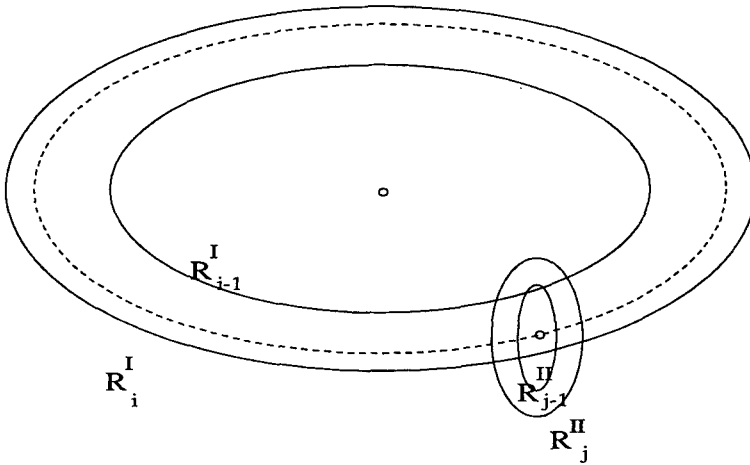


Fig. 4: The direct product of two annular blocks

Note that each block in the collection π is equal to the direct product of two annular blocks, see Figure 4.

Using the analogy with the annular partition for the double integrator system ([5]), it can be verified that π is an HIBC partition of D , i.e. each block X_{ij} is controllable.

Any two adjacent blocks $X_{i_1j_1}, X_{i_2j_2}$, i.e. such that $\overline{X}_{i_1j_1} \cap \overline{X}_{i_2j_2} \neq \emptyset$, are mutually dynamically consistent since motions arbitrarily close to radial motions are possible within and between blocks.

Let X, Y be two adjacent blocks of a f.a. partition $\sigma(D)$. Consider a piecewise analytic deformation σ' of σ such that

- (i) $\overline{X'} \cup \overline{Y'} = \overline{X} \cup \overline{Y}$;
- (ii) $X \subset X'$ and $Y' \subset Y$;
- (iii) for any block $Z \in \sigma$, such that $Z \neq X, Y$, $Z' = Z$.

We shall call such a deformation a *basic deformation*.

Let σ' be a basic deformation of a f.a. partition σ and let σ' be such that, for a given differential control system S satisfying the standard conditions, the distinguished blocks X, Y satisfy

- (i) for any state $y \in Y - \overline{Y'}$ there exists a controlled path of S passing from a state in X to y and from y into X which does not meet the deformed boundary;
- (ii) for any two distinct states in Y' which are connected by a controlled path of S meeting the deformed boundary there exists a controlled path connecting these states which lies in Y' .

We shall call such a deformation a *control paths dependent deformation*.

One can now prove that any control paths dependent deformation $\sigma'(D)$ of an HIBC partition $\sigma(D)$ is an HIBC partition of the state space D . Moreover, if control paths dependent deformations have been applied to σ a finite number of times, then the resulting deformation stays within the class of HIBC partitions of D .

In the particular case of the fully actuated Acrobot it may be verified that such control paths dependent deformations π' exist for the toroidal partition π described above. This follows from the existence of control paths dependent deformations for the annular partition for the double integrator system and from the fact that each block $X_{ij} \in \pi$ can be represented as the direct product of two annular blocks. But any two adjacent blocks of π are dynamically consistent and so we can conclude from an application of Theorem 3.3 that the same holds for any partition π^* obtained after a finite number of control paths dependent deformations.

5 Conditions Ensuring the HIBC Property of a Partition

Given a differential control system S satisfying the standard conditions (with $q = 1$), we define a (*continuous*) *positive fountain* to be a state such that (i) there exists a neighborhood of x such that all open ball neighborhoods of x which it contains are such that the accessibility set (for all finite times) from x relative to each neighborhood (i.e. via system trajectories contained in each neighborhood) is open when the state x is deleted, and (ii) the radius of the largest open ball neighborhood of x for which (i) holds is continuous in x .

A (*continuous*) *negative fountain* is a state which satisfies the analogous property to that above with coaccessibility replacing accessibility. Finally, a (*continuous*) *fountain* is state which is both a positive and a negative (continuous) fountain. Henceforth the adjective continuous will be dropped from the term fountain, in other words all fountains will be assumed to be continuous.

It is proved in [2] that if the open connected state space X of a differential control system is such that (i) every state is a fountain, and (ii) for every state $x \in X$ there exists a control function u_x such that x lies in the positive omega set of itself (i.e. x lies in the closure of the forward trajectory from x under u_x), then the state space X is controllable. A special case of this result is that where every state lies on some nontrivial controlled closed orbit in X .

We see that by an application of this result a finite analytic partition where each block satisfies the conditions (i) and (ii) is a HIBC partition.

Consider next Hamiltonian systems of the form

$$\dot{q} = \partial H / \partial p,$$

$$\dot{p} = -\partial H / \partial q + u,$$

where H is twice continuously differentiable. Define an *energy slice* for such a system as a connected subset of the state space for which the values of the Hamiltonian range over a specified open interval of the real line. Then it is shown in [2] that a Hamiltonian system where (i) all equilibria of the system under $u = 0$ are isolated, and (ii) all states are fountains, is such that any energy slice with compact closure is controllable.

An application of this result to the fully actuated Acrobot problem permits the construction of HIBC partitions, however the utility of blocks specified as energy slices is yet to be evaluated.

Acknowledgements The authors gratefully acknowledge conversations with Robert Hermann concerning parts of this work.

References

- [1] M. Broucke. A geometric theory of bisimulation of hybrid systems. Technical report, University of California at Berkeley, CA, October, 1997.
- [2] P.E. Caines and E. Lemch. Fountains, orbits and the global controllability of non-linear systems. Technical report, McGill University, Montreal, PQ, Canada, in preparation, 1998.
- [3] P.E. Caines and Y-J. Wei. On dynamically consistent hybrid systems. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *1994 Cornell University Workshop on Hybrid Systems & Autonomous Control, Hybrid Systems II*, LCNS 999, pages 257–263. Springer Verlag, NYC, 1995.
- [4] P.E. Caines and Y-J Wei. Hierarchical hybrid control systems. In S. Morse, editor, *Control Using Logic-Based Switching*, Volume 222, LNCIS, pages 1–12. Springer-Verlag, 1996.
- [5] P.E. Caines and Y-J. Wei. Hierarchical hybrid control systems: A lattice theoretic approach. *IEEE Transactions on Automatic Control*. To appear, 1998.
- [6] P.E. Caines and Y-J. Wei. The hierarchical lattices of a finite machine. *Systems and Control Letters*, 25:257–263, 95.
- [7] G. Lafferriere, G. J. Pappas, and S. Sastry. Subanalytic stratifications and bisimulations. Technical report, University of California at Berkeley, CA, January, 1998.
- [8] E. Lemch and P. E. Caines. On the existence of hybrid models for finite state machines. Submitted to *Systems and Control Letters*, 1997.
- [9] E. Lemch and P. E. Caines. Partition deformations of hierarchical hybrid control systems. Technical report, McGill University, Montreal, PQ, Canada, 1998.
- [10] M. W. Spong. Partial feedback linearization of underactuated mechanical systems. In *Proceedings, IROS'94*, pages 314–321, Munich, Germany, 1994.
- [11] M. W. Spong. The swing up control problem for the acrobot. *IEEE Control Systems*, 15(1):49–55, February, 1995.

Formal Verification of Safety-Critical Hybrid Systems*

Carolos Livadas and Nancy A. Lynch

Laboratory for Computer Science
Massachusetts Institute of Technology
{clivadas,lynch}@theory.lcs.mit.edu

Abstract. This paper investigates how formal techniques can be used for the analysis and verification of hybrid systems [1, 5, 7, 16] — systems involving both discrete and continuous behavior. The motivation behind such research lies in the inherent similarity of the hierarchical and decentralized control strategies of hybrid systems and the communication and operation protocols used for distributed systems in computer science. This paper focuses on the use of hybrid I/O automata [11, 12] to model, analyze, and verify safety-critical hybrid systems that use emergency control subsystems to prevent the violation of their safety requirements. The paper is split into two parts. First, we develop an abstract model of a *protector* — an emergency control component that guarantees that the physical plant at hand adheres to a particular safety requirement. The abstract protector model specialized to a particular physical plant and a particular safety requirement constitutes the specification of a protector that enforces the particular safety property for the particular physical plant. The correctness proof of the abstract protector model leads to simple correctness proofs of the implementations of particular protectors. In addition, the composition of independent protectors, and even dependent protectors under mild conditions, guarantees the conjunction of the safety properties guaranteed by the individual protectors being composed. Second, as a case study, we specialize the aforementioned abstract protector model to simplified versions of the personal rapid transit system (PRT 2000TM) under development at Raytheon Corporation and verify the correctness of overspeed and collision avoidance protectors. Such correctness proofs are repeated for track topologies ranging from a single track to a directed graph of tracks involving Y-shaped merges and diverges.

* This research was performed at the Theory of Distributed Systems Group of the Laboratory for Computer Science of the Massachusetts Institute of Technology. The research was supported by NSF Grant 9225124-CCR, U.S. Department of Transportation Contract DTRS95G-0001-YR.8, AFOSR-ONR Contracts F49620-94-1-0199 and F49620-97-1-0337, and ARPA Contracts N00014-92-J-4033 and F19628-95-C-0118.

1 Introduction

The recent trend of system integration and automation has encouraged the study of hybrid systems — systems that combine continuous and discrete behavior. Although the individual problems of continuous and discrete behavior have been extensively analyzed by control theory and formal analysis, respectively, their combination has recently been aggressively studied. In particular, the automation in various safety-critical systems, such as automated transportation systems, has indicated the need for formal approaches to system analysis, design, and verification. Automated highway systems [2,8], personal rapid transit systems [6,17], and air traffic control systems [9,15] have served as benchmark problems for the development of techniques to analyze, design, and verify hybrid systems.

Many of the safety-critical systems in use today abide by the engineering paradigm of using an emergency control, or protection, subsystem to prevent the violation of the system's safety requirements. In this paper we present a formal framework for the analysis of systems that adhere to this engineering paradigm. The framework is used to prove the correctness of such protection subsystems in an effort to provide indisputable system safety guarantees. The formal approach to the analysis of such systems has several advantages. Formal analysis yields a precise specification of the system and its safety requirements, provides insight as to the location of possible design errors, and minimizes the duplication of verification effort when such errors are corrected. The technique of system validation through exhaustive testing lacks the insightful feedback and requires full-fledged regression testing when design errors are detected.

In this paper, we use *hybrid I/O automata* [11,12] — an extension of *timed I/O automata* [4,14] — to define an abstract model of a *protector* — a subsystem that guarantees that the physical plant adheres to a particular safety requirement. The abstract protector model is parameterized in terms of the physical plant, the safety requirement, and several other quantities. The instantiation of the abstract protector, obtained by specifying the abstract protector's parameters, constitutes the specification of a protector that guarantees a particular safety property for a particular physical plant model. The proof of correctness of the abstract protector model minimizes the effort in verifying the correct operation of a particular protector implementation. In fact, such correctness proofs get reduced to simple simulations from the protector implementations to the particular instantiation of the abstract protector model. As a case study, we apply the formal framework developed towards the verification of overspeed and collision protection subsystems for simplified models of the personal rapid transit system (PRT 2000™) under development at Raytheon Corporation. The case studies presented in this paper extend the work of Weinberg, Lynch, and Delisle [17] by introducing a powerful formal framework that allows more complete system models to be used. The actual PRT 2000™ system is comprised of 4-passenger vehicles that travel on an elevated guideway of tracks involving Y-shaped merges and diverges and provide point-to-point passenger transportation. In this treatment, we verify the correct operation of overspeed and collision avoidance protectors for track topologies ranging from a single track to a directed

graph of tracks involving Y-shaped merges and diverges. A detailed treatment of the work presented in this paper can be found in Ref. 6.

2 Hybrid I/O Automata

A hybrid I/O automaton A is a (possibly) infinite state model of a system involving both discrete and continuous behavior. The automaton $A = (U, X, Y, \Sigma^{in}, \Sigma^{int}, \Sigma^{out}, \Theta, \mathcal{D}, \mathcal{W})$ consists of three disjoint sets U , X , and Y of variables (*input*, *internal*, and *output* variables, respectively), three disjoint sets Σ^{in} , Σ^{int} , and Σ^{out} of *actions* (*input*, *internal*, and *output* actions, respectively), a non-empty set Θ of *initial states*, a set \mathcal{D} of *discrete transitions* and a set \mathcal{W} of trajectories over V , where $\Sigma = \Sigma^{in} \cup \Sigma^{int} \cup \Sigma^{out}$ and $V = U \cup X \cup Y$. The initial states, the discrete transitions, and the trajectories of any HIOA A must however satisfy several technical conditions which are omitted here. For a detailed presentation of the HIOA model, the reader is referred to Refs. 11 and 12.

Variables in the set V are typed; that is, each variable $v \in V$ ranges over the set of values $type(v)$. A *valuation* of V , also referred to as a *state* of A , is a function that associates to each variable v of V a value in $type(v)$. The set of all valuations of V , or equivalently the set of all states of A , is denoted by \mathbf{V} , or equivalently $states(A)$. Letting $v \in V$ and $S_v \subseteq type(v)$, we use the notation $v : \in S_v$ to denote the assignment of an arbitrary element of the set S_v to the variable v . Similarly, letting $S_V \subseteq \mathbf{V}$, we use the notation $V : \in S_V$ to denote the assignment of an element of the set $type(v)$ to the variable v , for each v in V , such that the resulting valuation of V is an arbitrary element of the set S_V . Letting s be a state of A , i.e., $s \in \mathbf{V}$, and $V' \subseteq V$, we define the *restriction* of s to V' , denoted by $s[V']$, to be the valuation s' of the variables of V' in s . Letting $\mathbf{X} \subseteq \mathbf{V}$, we say that \mathbf{X} is *V' -determinable* if for all $x \in \mathbf{X}$ and $s \in \mathbf{V}$, such that $x[V'] = s[V']$, it is the case that $s \in \mathbf{X}$. The continuous time evolution of the valuations of the variables in V is described by a *trajectory* w over V ; that is, a function $T_I \rightarrow \mathbf{V}$, where T_I is a left-closed interval of $\mathbb{R}^{\geq 0}$ with left endpoint equal to 0. The *limit time* of w , denoted by $w.ltime$, is defined to be the supremum of the domain of w , $dom(w)$. We define the *first state* of a trajectory w , denoted by $w.fstate$, to be the state $w(0)$. Moreover, if the domain of a trajectory w is right-closed, then we define the *last state* of w , denoted by $w.lstate$, to be the state $w(w.ltime)$.

A *hybrid execution fragment* α of A is a finite or infinite alternating sequence $w_0 a_1 w_1 a_2 w_2 \dots$, where $w_i \in \mathcal{W}$, $a_i \in \Sigma$, and if w_i is not the last trajectory of α then w_i is right-closed and the discrete transition $(w_i.lstate, a_{i+1}, w_{i+1}.fstate)$ is in \mathcal{D} , or equivalently $w_i.lstate \xrightarrow{a_{i+1}} w_{i+1}.fstate$. If $w_0.fstate \in \Theta$ then α is a *hybrid execution* of A . A hybrid execution α of A is *finite* if it is a finite sequence and the domain of its final trajectory is a right-closed interval and *admissible* if $\alpha.ltime = \infty$. If $R \subseteq states(A)$ and $s, s' \in R$, then s' is *R -reachable from s* provided that there is a hybrid execution fragment of A that starts in s , ends in s' , and all of whose states are in the set R .

The *hybrid trace* of a hybrid execution fragment α of A , denoted by $h\text{-trace}(\alpha)$, is the sequence obtained by projecting α onto the external variables of A and subsequently removing all inert internal and environment actions. The set of *hybrid traces* of A , denoted by $h\text{-traces}(A)$, is the set of hybrid traces that arise from all the finite and admissible hybrid executions of A .

A *superdense time* in an execution fragment α of A is a pair (i, t) , where $t \leq w_i.ltime$. We totally order superdense times in the execution fragment α lexicographically. An *occurrence* of a state s in an execution fragment α of A is a triple (i, t, s) such that (i, t) is a superdense time in α and $s = w_i(t)$. State occurrences in α are ordered according to their superdense times. If S is a set of states of A and α is an execution fragment of A , then $past(S, \alpha)$ is the set of state occurrences (i, t, s) in α such that either $s \in S$ or there is a previous state occurrence (i', t', s') in α with $s' \in S$.

Two HIOA A_1 and A_2 are *compatible* if $X_i \cap V_j = Y_i \cap Y_j = \Sigma_i^{int} \cap \Sigma_j = \Sigma_i^{out} \cap \Sigma_j^{out} = \emptyset$, for $i, j \in \{1, 2\}, i \neq j$. If A_1 and A_2 are compatible then their *composition* $A_1 \times A_2$ is defined to be the tuple $A = (U, X, Y, \Sigma^{in}, \Sigma^{int}, \Sigma^{out}, \Theta, \mathcal{D}, \mathcal{W})$ given by $U = (U_1 \cup U_2) - (Y_1 \cup Y_2)$, $X = X_1 \cup X_2$, $Y = Y_1 \cup Y_2$, $\Sigma^{in} = (\Sigma_1^{in} \cup \Sigma_2^{in}) - (\Sigma_1^{out} \cup \Sigma_2^{out})$, $\Sigma^{int} = \Sigma_1^{int} \cup \Sigma_2^{int}$, $\Sigma^{out} = \Sigma_1^{out} \cup \Sigma_2^{out}$, $\Theta = \{s \in V \mid s[V_1 \in \Theta_1 \wedge s[V_2 \in \Theta_2]\}$ and sets of discrete transitions \mathcal{D} and trajectories \mathcal{W} each of whose elements projects to discrete transitions and trajectories, respectively, of A_1 and A_2 .

Two HIOA A_1 and A_2 are *comparable* if they have the same external interface, i.e., $U_1 = U_2$, $Y_1 = Y_2$, $\Sigma_1^{in} = \Sigma_2^{in}$, and $\Sigma_1^{out} = \Sigma_2^{out}$. If A_1 and A_2 are comparable, then $A_1 \leq A_2$ is defined to denote that the hybrid traces of A_1 are included in those of A_2 ; that is, $A_1 \leq A_2 \triangleq h\text{-traces}(A_1) \subseteq h\text{-traces}(A_2)$. If $A_1 \leq A_2$, then we say that A_1 *implements* A_2 .

3 Protected Plant Systems

A *protected plant system* is modeled abstractly as a *physical plant* interacting with a *protection system*. The protection system is modeled as the composition of a set of *protectors* each of which is supposed to enforce a particular safety requirement of the physical plant. Our model is abstract in the sense that it does not specify any of the details or safety requirements of the physical plant.

The physical plant and each of the protectors are modeled as HIOA. The *physical plant* PP is an automaton that is assumed to be interacting with the protectors through the set J of communication channels, which are referred to as *ports*. The input action set Σ_{PP}^{in} , the output action set Σ_{PP}^{out} , and the input variable set U_{PP} are partitioned into subsets $\Sigma_{PP_j}^{in}$, $\Sigma_{PP_j}^{out}$, and U_{PP_j} , respectively, one for each port j . We use the letter p to denote a state of PP and P to denote a set of states of PP . A *protector* A for the physical plant PP and the port set $K \subseteq J$ is an automaton that is compatible with PP and whose output actions are exactly the input actions of PP on ports in K , whose output variables are exactly the input variables of PP on ports in K , and all of whose input actions and input

variables are outputs of PP . It can easily be shown that the composition of two distinct protectors is itself a protector.

Letting S , R , and G be particular sets of states of PP , a protector automaton A for PP and ports K *guarantees* G in PP from S given R provided that every finite execution of the composition $PP \times A$ starting in a state in S that only involves states in R ends in a state in G regardless of the inputs that arrive at PP on ports other than those in K . Two protectors are *dependent*, if the correct operation of one relies on the correct operation of the other, and *independent*, otherwise. The following theorems express the *substitutivity* condition — the condition under which the implementation of a protector is *correct* with respect to its specification — and the *compositional* conditions — conditions under which the composition of independent or dependent protectors guarantees the conjunction of the safety properties guaranteed by the protectors being composed.

Theorem 1 (Substitutivity). *Let A_1 and A_2 be two protector automata for the same port set K of a physical plant automaton PP , and suppose that $A_1 \leq A_2$. If A_2 guarantees G in PP from S given R , then A_1 guarantees G in PP from S given R .*

Theorem 2 (Independent Protector Composition). *Suppose that A_1, A_2, \dots, A_k are protector automata for a physical plant automaton PP , with respective port sets K_1, K_2, \dots, K_k , where $K_i \cap K_{i'} = \emptyset$, for all $i, i' \in \{1, \dots, k\}, i \neq i'$. Suppose that each of the protectors A_i , for all $i \in \{1, \dots, k\}$, guarantees G_i from S_i given R_i . If the protectors A_1, A_2, \dots, A_k are compatible, then their composition $\prod_{i \in \{1, \dots, k\}} A_i$ is a protector for PP that guarantees $\bigcap_{i \in \{1, \dots, k\}} G_i$ from $\bigcap_{i \in \{1, \dots, k\}} S_i$ given $\bigcap_{i \in \{1, \dots, k\}} R_i$.*

Theorem 3 (Dependent Protector Composition). *Suppose that A_1, A_2, \dots, A_k are protector automata for a physical plant automaton PP , with respective port sets K_1, K_2, \dots, K_k , where $K_i \cap K_{i'} = \emptyset$, for all $i, i' \in \{1, \dots, k\}, i \neq i'$. Suppose that each of the protector automata A_i , for all $i \in \{1, \dots, k\}$, guarantees G_i from S_i given $R_i \cap \left(\bigcap_{i' \in \{1, \dots, k\}, i' \neq i} G_{i'} \right)$.*

Assume that α is any finite execution of the system $PP \times \prod_{i \in \{1, \dots, k\}} A_i$ starting from a state in $\bigcap_{i \in \{1, \dots, k\}} S_i$ and all of whose states are in the set $\bigcap_{i \in \{1, \dots, k\}} R_i$. Then, one of the following holds:

1. *Every state in α is in $\bigcap_{i \in \{1, \dots, k\}} G_i$.*
2. *The finite execution α can be written as $\alpha_1 \frown \alpha_2$, where*
 - (a) *all state occurrences in α_1 , except possibly the last, are in the set of states $\bigcap_{i \in \{1, \dots, k\}} G_i$,*
 - (b) *if the last state occurrence in α_1 is in $\overline{G_i}$, for some $i \in \{1, \dots, k\}$, then there exists $i' \in \{1, \dots, k\}, i' \neq i$, such that the last state occurrence in α_1 is in $\overline{G_{i'}}$, and*
 - (c) *all state occurrences in α_2 , except possibly the first, are in the set of states $\bigcap_{i \in I} \text{past}(\overline{G_i}, \alpha)$, for some $I \subseteq \{1, \dots, k\}$, where $|I| \geq 2$.*

In loose terms, Theorem 3 states that the composition of dependent protectors guarantees the conjunction of the safety properties guaranteed by the protectors being composed provided a single action or trajectory of the composed system can cause the violation of *at most* one of the safety properties guaranteed by the protectors being composed.

4 An Abstract Protector

The abstract protector automaton is parameterized in terms of the automaton PP , the subsets R , G , and S of the states of PP , the port index j , and the positive real-valued sampling period d . The PP automaton represents the physical plant being modeled. The set R , also referred to as the set of *reliance*, is the set of states to which we restrict the states of the PP automaton while considering a particular protector. This set is usually comprised of states satisfying a particular property of the physical plant that is required by the protector under consideration. The set G , also referred to as the set of *guarantee*, is the set of states to which the protector is designed to constrain the PP automaton. The set S is a set of states from which the protector under consideration is said to guarantee G given R ; that is, given that the states of the PP automaton are restricted to the set R , the protector guarantees that every finite execution starting from an initial state in S ends in a state in G . The port index j and the sampling period d denote the port and the sampling period with which the abstract protector interacts with the PP automaton. Thus, an instantiation of the abstract protector automaton $Abs(PP, S, R, G, j, d)$ is obtained by specifying the parameters PP , *etc.*

To begin, we define several functions and sets that are useful in the definition of the abstract protector $Abs(PP, S, R, G, j, d)$. Although, formal definitions of these functions and sets are presented in Table 1, their informal interpretations follow. First, we define a function, $future_{PP,R,j}$, that yields the set of states of PP that are R -reachable from the given subset of R within an amount of time in the given subset of $\mathbb{R}^{\geq 0}$, under the constraint that no input actions arrive on port j of the PP automaton. We define a function, $no-op_{PP,R,j}$, which yields, for a given state in R , the set of input actions on port j of the PP automaton that do not affect the state of the PP automaton, provided they are executed prior to either time-passage, or other input actions on port j . For any state p in R , the input actions in the set $no-op_{PP,R,j}(p)$ are referred to as no-op input actions on port j of PP for the state p . We define a set, $very-safe_{PP,R,G,j}$, which is comprised of the states of PP that satisfy R and from which all R -reachable states of PP with no input actions on port j are in G . The set $very-safe_{PP,R,G,j}$ may be interpreted as the set consisting of the states from which the PP automaton is bound to remain within the set G provided that it remains within the set R and the protector on port j does not retract or issue additional protective actions. We define a set, $safe_{PP,R,G,j}$, which is comprised of the states of PP that satisfy R and from which the protector on port j has a "winning protective strategy"; that is, for any state p in $safe_{PP,R,G,j}$ there exists an input action on port j of the PP automaton whose immediate execution — its execution prior to any

Table 1 Terminology for the abstract protector $Abs(PP, S, R, G, j, d)$.

$future_{PP,R,j} : \mathcal{P}(R) \times \mathcal{P}(\mathbb{R}^{\geq 0}) \rightarrow \mathcal{P}(R)$, defined by:

$p \in future_{PP,R,j}(P, T)$, where $P \subseteq R$ and $T \subseteq \mathbb{R}^{\geq 0}$, if and only if p is R -reachable from some $p' \in P$ via a finite execution fragment α of PP with no input actions on port j and with $\alpha.time \in T$.

$no-op_{PP,R,j} : R \rightarrow \mathcal{P}(\Sigma_{PP,j}^{in})$, defined by:

$\pi \in no-op_{PP,R,j}(p)$ if and only if π is an input action on port j of PP such that for all $p', p'' \in R$ satisfying $p' \in future_{PP,R,j}(p, 0)$ and $p' \xrightarrow{\pi}_{PP} p''$, it is the case that $p'' = p'$.

$very-safe_{PP,R,G,j} \subseteq R$, defined by:

$p \in very-safe_{PP,R,G,j}$ if and only if $future_{PP,R,j}(p, \mathbb{R}^{\geq 0}) \subseteq G$.

$safe_{PP,R,G,j} \subseteq R$, defined by:

$p \in safe_{PP,R,G,j}$ if and only if both of the following hold:

1. $future_{PP,R,j}(p, 0) \subseteq G$.
2. There exists an input action π on port j , such that for every $p', p'' \in R$ satisfying $p' \in future_{PP,R,j}(p, 0)$ and $p' \xrightarrow{\pi}_{PP} p''$, it is the case that $p'' \in very-safe_{PP,R,G,j}$.

$safe_{PP,R,G,j} : \Sigma_{PP,j}^{in} \rightarrow \mathcal{P}(R)$, defined by:

$p \in safe_{PP,R,G,j}(\pi)$ if and only if both of the following hold:

1. $future_{PP,R,j}(p, 0) \subseteq G$.
2. For every $p', p'' \in R$ such that $p' \in future_{PP,R,j}(p, 0)$ and $p' \xrightarrow{\pi}_{PP} p''$, it is the case that $p'' \in very-safe_{PP,R,G,j}$.

$delay-safe_{PP,R,G,j} : \mathbb{R}^{\geq 0} \rightarrow \mathcal{P}(R)$, defined by:

$p \in delay-safe_{PP,R,G,j}(t)$ if and only if both of the following hold:

1. $future_{PP,R,j}(p, [0, t]) \subseteq G$.
2. $future_{PP,R,j}(p, t) \subseteq safe_{PP,R,G,j}$.

time-passage with the possibility that its execution follows an arbitrary number of discrete actions other than input actions on port j — guarantees that all subsequent R -reachable states of PP with no input actions on port j are in G ; that is, the state following the execution of the particular input action of PP on port j is in the set $very-safe_{PP,R,G,j}$. We overload the notation $safe_{PP,R,G,j}$ by defining a function, $safe_{PP,R,G,j}$, which yields the states of PP that satisfy R and for which the immediate execution of the given input action on port j — its execution prior to any time-passage with the possibility that its execution follows an arbitrary number of discrete actions other than input actions on port j — guarantees that all subsequent R -reachable states of PP with no input actions on port j are in G ; that is, the state following the execution of the given input action on port j is in the set $very-safe_{PP,R,G,j}$. Finally, we define a function, $delay-safe_{PP,R,G,j}$, which yields the set of states of PP that satisfy R and for which all states R -reachable within the given amount of time and with no input actions on port j are in G , and all states R -reachable in exactly the given amount of time and with no input actions on port j are in $safe_{PP,R,G,j}$.

We proceed by stating the various assumptions made about the physical plant PP and the abstract protector $Abs(PP, S, R, G, j, d)$. We assume that the

Fig. 1 *Sensor*(PP, S, R, G, j, d) automaton definition.

Actions:	Input:	e , the environment action
	Output:	$\text{snapshot}(y)_j$, for each valuation y of Y_{PP}
Variables:	Input:	$u \in \text{type}(u)$, for all $u \in Y_{PP}$, initially $u \in \text{type}(u)$, for each $u \in Y_{PP}$
	Internal:	$\text{now}_j \in \mathbb{R}^{\geq 0}$, initially 0 $\text{next-snap}_j \in \mathbb{R}^{\geq 0}$, initially 0
Discrete Transitions:		
e		
	Eff: $Y_{PP} : \in Y_{PP}$	$\text{snapshot}(y)_j$ Pre: $\text{next-snap}_j = \text{now}_j$ y is current valuation of Y_{PP} Eff: $Y_{PP} : \in Y_{PP}$ $\text{next-snap}_j := \text{now}_j + d$
Trajectories:		
for all $u \in Y_{PP}$		
u assumes arbitrary values in $\text{type}(u)$ throughout w		
next-snap_j is constant throughout w		
for all $t \in T_I$		
$w(t).\text{now}_j = w(0).\text{now}_j + t$		
$w(t).\text{now}_j \leq w(t).\text{next-snap}_j$		

PP automaton has no input variables on port j , for all $j \in J$; that is, the protectors control the state of the physical plant only through input actions. A consequence of this assumption is that the environment action of the PP automaton is stuttering. Moreover, we assume that the PP automaton has no output actions on port j , for all $j \in J$. The physical plant is modeled as a passive system in the sense that the protectors observe the state of the plant only through output variables. We assume that there exist no-op input actions on port j for every state of the PP automaton in the set R . We assume that membership of a state of the PP automaton in the set $\text{safe}_{PP,R,G,j}$ is determinable from the output variables of the PP automaton, i.e., the set $\text{safe}_{PP,R,G,j}$ is Y_{PP} -determinable. Moreover, we assume that for any state in the set $\text{safe}_{PP,R,G,j}$, an appropriate action to guarantee safety can be determined from the output variables of the PP automaton, i.e., the variables in Y_{PP} . For any valuation y of the output variables Y_{PP} of the PP automaton, we use the notation $y \in \text{safe}_{PP,R,G,j}$ to denote the existence of a state $p \in \text{safe}_{PP,R,G,j}$ such that $p[Y_{PP} = y]$. We assume that the state information provided by the output variables of the PP automaton is sufficient to determine membership of any state of the PP automaton in the sets R and G , i.e., the sets R and G are Y_{PP} -determinable. Moreover, we assume that the set of start states S is a subset of the set $\text{safe}_{PP,R,G,j}$.

The protector is defined as the composition of a *sensor automaton* (Figure 1) and a *discrete controller automaton* (Figure 2). Both the sensor and the discrete controller are described abstractly in terms of PP, S, R, G, j , and d and are respectively denoted $\text{Sensor}(PP, S, R, G, j, d)$ and $\text{DC}(PP, S, R, G, j, d)$. At intervals of d time units, the sensor automaton samples the output variables of the PP automaton. The discrete controller automaton is rather nondeterminis-

Fig. 2 $DC(PP, S, R, G, j, d)$ automaton definition.

Actions:	Input:	e , the environment action (stuttering) $\text{snapshot}(y)_j$, for each valuation y of Y_{PP}
	Output:	π , for all $\pi \in \Sigma_{PP}^{\text{in}}$
Variables:	Internal:	$\text{send}_j \in \Sigma_{PP}^{\text{in}} \cup \{\text{null}\}$, initially null
Discrete Transitions:		
e		$\text{snapshot}(y)_j$
	Eff: None	Eff: if $y \in \text{safe}_{PP,R,G,j}$ then $\text{send}_j := \{\phi \in \Sigma_{PP}^{\text{in}} \mid$ $\forall p, p', p'' \in R \text{ such that}$ $p \models Y_{PP} = y, p' \in \text{future}_{PP,R,j}(p, 0),$ $\text{and } p' \xrightarrow{\phi}_{PP} p'',$ $\text{it is the case that}$ $p'' \in \text{delay-safe}_{PP,R,G,j}(d)\}$
π	Pre: $\text{send}_j = \pi$ Eff: $\text{send}_j := \text{null}$	$\text{send}_j := \Sigma_{PP}^{\text{in}}$
		else $\text{send}_j := \Sigma_{PP}^{\text{in}}$
Trajectories:		
		$w.\text{send}_j \equiv \text{null}$

tic. Based on the output state information of the PP automaton sampled by the sensor automaton, the discrete controller automaton issues protective actions so as to guarantee that (i) the PP automaton remains within the set G up to the next sampling point, and (ii) the state of the PP automaton at the next sampling point is in the set $\text{safe}_{PP,R,G,j}$. The nondeterminism in the description of the $DC(PP, S, R, G, j, d)$ automaton allows the freedom to choose any response that satisfies the given conditions — however, in a discrete controller automaton implementation, a response that least restricts the future states of the physical plant automaton PP would be preferred because it would represent a weaker protective action.

Theorem 4. $\text{Abs}(PP, S, R, G, j, d)$ guarantees G in PP from S given R .

The correctness proof of a particular protector implementation involves defining the particular protector's specification as the instantiation of the abstract protector for particular definitions of PP , etc. and showing that the particular protector implementation is correct with respect to the particular instantiation of the abstract protector. The first step simply involves specifying the parameters PP , etc. The second step is simplified by choosing the protector implementation to be the composition of the sensor automaton $\text{Sensor}(PP, S, R, G, j, d)$ and a discrete automaton that is chosen so as to guarantee the effect clause of the $\text{snapshot}(y)_j$ action in $DC(PP, S, R, G, j, d)$. Thus, the correctness proof of the implementation is reduced to a simulation from the implementation of the discrete controller automaton to its specification.

5 Modeling the PRT 2000™

In this section, we present a model for a simplified version of the PRT 2000™ whose track topology involves a single track. The model, **VEHICLES**, which is presented in Figure 3, is a HIOA that conforms to the restrictions and assumptions made about the *PP* automaton in Sections 3 and 4. It involves n vehicles of identical dimensions and acceleration/deceleration capabilities traveling on a single track. Its state variables include the position x_i , the velocity \dot{x}_i , and the acceleration \ddot{x}_i of each vehicle i in the set of vehicles I and several other variables that record whether each vehicle has collided *into* each other vehicle (*collided*(i, i'), for $i' \in I, i' \neq i$), whether each vehicle is braking (*brake*(i), for $i \in I$), and whether each protector j in the set of protectors J is requesting each particular vehicle to brake (*brake-req*(i, j), for $i \in I$ and $j \in J$). Several properties of the physical plant are enforced by restricting the states of the **VEHICLES** automaton to the set *VALID* (Appendix A). In particular, we assume that the vehicles occupy non-overlapping sections of the track, the vehicles are only allowed to move forward on the track, the non-malfunctioning vehicle acceleration/deceleration capabilities to be within the interval $[\ddot{c}_{min}, \ddot{c}_{max}]$, and the non-malfunctioning braking deceleration to be given by \ddot{c}_{brake} , if the vehicle is moving forward, and 0, otherwise.

The formal definitions of the derived variables and sets of the **VEHICLES** automaton are shown in Appendix A. For brevity, we only give informal definitions of the key derived variables. Each of the variables E_i , for $i \in I$, denotes the *extent* of the vehicle i ; that is, the section of the track *occupied* by the vehicle i . It is defined as the section of track ranging from the position of the rear of the vehicle i to the point on the track that is a distance of c_{len} downstream of the rear of the vehicle i — a distance that specifies the minimum allowable separation between vehicles, *i.e.*, $E_i = [x_i, x_i + c_{len}]$, for $i \in I$. Each of the variables O_i , for $i \in I$, denotes the section of the track that the vehicle i *owns*; that is, the range extending from the current position of the rear of the vehicle i to the point on the track that the vehicle can reach even if it is braked immediately. Each of the variables $C_i(t)$, for $i \in I$ and $t \in \mathbb{R}^{\geq 0}$, denotes the section of the track that the vehicle i *claims* within t time units; that is, the range extending from the current position of the rear of the vehicle i to the point on the track that the vehicle i can reach if it is braked after t time units and assuming worst-case vehicle behavior up to the point in time when it is braked. Moreover, each of the variables *collided*($*, i, *$), for $i \in I$, denotes whether the vehicle i has ever been involved in a collision. Some auxiliary sets for the vehicles automaton that will be used in the following sections are defined in Appendix B.

The input actions of the **VEHICLES** automaton are the environment action e and the actions *brake*(i) _{j} and *unbrake*(i) _{j} , for $i \in I$ and $j \in J$. Since the **VEHICLES** automaton has no input variables, the environment action e is stuttering. Each of the actions *brake*(i) _{j} and *unbrake*(i) _{j} , for $i \in I$ and $j \in J$, correspond to actions performed by the protector j instructing the vehicle i to apply or release its “emergency” brake, respectively. Each *brick-wall*(i) action, for $i \in I$, models the instantaneous stopping of the vehicle i — as if it hit a brick wall.

Fig. 3 The VEHICLES automaton.

Actions:	Variables
Input:	Internal:
e , the environment action (stuttering) $brake(i)_j$, for all $i \in I, j \in J$ $unbrake(i)_j$, for all $i \in I, j \in J$	$\ddot{x}_i \in \mathbb{R}$, for all $i \in I$, initially $\ddot{x}_i \in \mathbb{R}$ $brake(i) \in \text{Bool}$, for all $i \in I$, initially False $brake\text{-}req(i, j) \in \text{Bool}$, for all $i \in I, j \in J$, initially False
Internal:	Output:
$colliding\text{-}pair(i, i')$, for all $i, i' \in I, i' \neq i$ $collision\text{-}effects(i)$, for all $i \in I$ $brick\text{-}wall(i)$, for all $i \in I$	$x_i \in \mathbb{R}$, for all $i \in I$, initially $x_i \in \mathbb{R}$ $\dot{x}_i \in \mathbb{R}$, for all $i \in I$, initially $\dot{x}_i \in \mathbb{R}$ $collided(i, i') \in \text{Bool}$, for all $i, i' \in I, i' \neq i$, initially False subject to <i>VALID</i>
Discrete Transitions:	
e	$colliding\text{-}pair(i, i')$ Pre: $\neg collided(i, i')$ $\wedge (E_i \cap E_{i'} \neq \emptyset)$ $\wedge (x_i < \min(E_i \cap E_{i'}))$ Eff: $collided(i, i') := \text{True}$
$brake(i)_j$	$collision\text{-}effects(i)$ Pre: $collided(*, i, *)$ Eff: $\dot{x}_i := \mathbb{R}^{\geq 0}$ $\ddot{x}_i := \mathbb{R}$
Eff: $brake\text{-}req(i, j) := \text{True}$ if $\neg brake(i)$ then $brake(i) := \text{True}$ if $\dot{x}_i = 0$ then $\ddot{x}_i := 0$ else $\ddot{x}_i := \ddot{c}_{brake}$	$brick\text{-}wall(i)$ Pre: True Eff: $\dot{x}_i := 0$ if $brake(i)$ then $\ddot{x}_i := 0$ else $\ddot{x}_i := [0, \ddot{c}_{max}]$
$unbrake(i)_j$	
Eff: $brake\text{-}req(i, j) := \text{False}$ if $brake(i)$ $\wedge (\neg \vee_{k \in J} brake\text{-}req(i, k))$ then $brake(i) := \text{False}$ $\ddot{x}_i := [\ddot{c}_{min}, \ddot{c}_{max}]$	
Trajectories:	
for all $i, i' \in I, i \neq i'$, $collided(i, i')$ is constant throughout w for all $i \in I$ and $j \in J$, $brake(i)$ and $brake\text{-}req(i, j)$ are constant throughout w for all $i, i' \in I, i \neq i'$ the function $w.\ddot{x}_i$ is integrable for all $t \in T_I$ $w(t).\dot{x}_i = w(0).\dot{x}_i + \int_0^t w(s).\ddot{x}_i ds$ $w(t).x_i = w(0).x_i + \int_0^t w(s).\dot{x}_i ds$ if $\neg w.collided(i, i')$ $\wedge (w(t).E_i \cap w(t).E_{i'} \neq \emptyset)$ $\wedge (w(t).x_i < \min(w(t).E_i \cap w(t).E_{i'}))$ then $t = w.time$ subject to <i>VALID</i>	

Thereafter however, the vehicle i is allowed to reinitiate forward motion. Each **colliding-pair**(i, i') action, for $i, i' \in I, i \neq i'$, records the fact that the vehicle i has collided into the vehicle i' . Since the trailing vehicle is the only vehicle that can prevent the collision through braking, a collision is recorded only by the trailing vehicle as if the trailing vehicle were the only vehicle liable for the particular collision. Each **collision-effects**(i) action, for $i \in I$, models the adverse effects of a collision involving the vehicle i and may be executed, even repeatedly, at any instant of time following the first collision involving the vehicle i . Thus, the malfunctioning apparatus of any vehicle i , for $i \in I$, is modeled by succeeding each of the discrete actions with a **collision-effects**(i) action for the malfunctioning vehicle.

The trajectories of the VEHICLES automaton model the continuous evolution of the state of the VEHICLES automaton. If during a trajectory a vehicle i collides into a vehicle i' for the first time, the trajectory is stopped so that the collision can be recorded.

6 Example Overspeed and Collision Avoidance Protectors

6.1 Example 1: Overspeed Protection System

In this section, we present a protector, called OS-PROT, that prevents the vehicles of the VEHICLES automaton from exceeding a prespecified global speed limit \dot{c}_{max} , provided that they do not collide among themselves. The protector OS-PROT is defined to be the composition of n separate copies of another protector called OS-PROT-SOLO $_i$, one copy for each vehicle $i \in I$. Each of the OS-PROT-SOLO $_i$ protectors, for $i \in I$, guarantees that the vehicle i , does not exceed the speed limit \dot{c}_{max} , provided that no collisions among any of the vehicles occur. The braking strategy of the OS-PROT-SOLO $_i$ protector is to instruct the vehicle i to brake if it is capable of exceeding the speed limit \dot{c}_{max} within the time until the next sampling point.

Let PP_i be the VEHICLES automaton of Figure 3, the port j_i and the sampling period d_i be the port and sampling period with which the protector OS-PROT-SOLO $_i$ communicates with the VEHICLES automaton, the set R_i be the set of states in which none of the vehicles have ever collided, i.e., $R_i = P_{not-collided}$ (Appendix B), the set G_i be the set of states in which the vehicle i is at or below the speed limit, i.e., $G_i = VALID - P_{overspeed(i)}$ (Appendix B), and the set S_i be the set $safe_{PP_i, R_i, G_i, j_i}$. We define the OS-PROT-SOLO $_i$ automaton to be the composition of $Sensor(PP_i, S_i, R_i, G_i, j_i, d_i)$ and the discrete controller automaton of Figure 4.

Lemma 1. *The protector OS-PROT-SOLO $_i$ guarantees G_i in VEHICLES starting from S_i given R_i .*

Corollary 1. *The protector OS-PROT = $\prod_{i \in I} OS-PROT-SOLO_i$ for the VEHICLES automaton guarantees $\bigcap_{i \in I} G_i$ in the VEHICLES automaton starting from $\bigcap_{i \in I} S_i$ given $P_{not-collided}$.*

Corollary 1 follows directly from Lemma 1 and Theorem 2.

Fig. 4 Discrete controller automaton for the protector OS-PROT-SOLO_{*i*}.

Actions:	Input:	e , the environment action (stuttering)
	Output:	$\text{snapshot}(y)_j$, for each valuation y of Y_{VEHICLES}
		$\text{brake}(i)_j$
		$\text{unbrake}(i)_j$
Variables:	Internal:	$\text{send}_j \in \{\text{brake}, \text{unbrake}, \text{null}\}$, initially null
Discrete Transitions:		
e		
	Eff: None	$\text{brake}(i)_j$ Pre: $\text{send}_j = \text{brake}$ Eff: $\text{send}_j := \text{null}$
$\text{snapshot}(y)_j$		
	Eff: if $(y.\hat{x}_i \leq \dot{c}_{\max} - d\ddot{c}_{\max})$ then	$\text{unbrake}(i)_j$ Pre: $\text{send}_j = \text{unbrake}$ Eff: $\text{send}_j := \text{null}$
	$\text{send}_j := \text{unbrake}$	
	else	
	$\text{send}_j := \text{brake}$	
Trajectories:		
	$w.\text{send}_j \equiv \text{null}$	

6.2 Example 2: Collision Avoidance on a Single Track

In this section, we present a protector, called CL-PROT, that prevents the vehicles of the VEHICLES automaton from colliding among themselves, provided that they are all abiding by the speed limit \dot{c}_{\max} . The protector CL-PROT is defined to be the composition of n separate copies of another protector called CL-PROT-SOLO_{*i*}, one copy for each vehicle $i \in I$. Each of the OS-PROT-SOLO_{*i*} protectors, for $i \in I$, guarantees that the vehicle i does not collide into any of the vehicles it trails, provided that all the vehicles in the VEHICLES automaton are abiding by the speed limit and that all other vehicles $i' \in I, i' \neq i$, do not collide into any of the vehicles they respectively trail. The braking strategy of the CL-PROT-SOLO_{*i*} protector is to instruct the vehicle i to brake if it has a d_i time unit claim overlap with any of the vehicles it trails. The rationale behind this braking strategy is that a collision between two vehicles in the VEHICLES automaton can only be prevented by instructing the trailing vehicle to brake.

Let PP_i be the VEHICLES automaton of Figure 3, the port j_i and the sampling period d_i be the port and sampling period with which the protector CL-PROT-SOLO_{*i*} communicates with the VEHICLES automaton, and the set G_i be the set of states in which the vehicle i has not collided into any of the other vehicles, i.e., $G = \text{VALID} - P_{\text{collided}(i)}$ (Appendix B). Moreover, let the set R_i be the set of states in which all of the vehicles are abiding by the speed limit and in which each of the other vehicles has never collided into any other vehicle, i.e., $R_i = P_{\text{not-overspeed}} \cap \left(\bigcap_{i' \in I, i' \neq i} G_{i'} \right)$ (Appendix B), and the set S_i be the set $\text{safe}_{PP_i, R_i, G_i, j_i}$. We define the CL-PROT-SOLO_{*i*} automaton to be the composition of $\text{Sensor}(PP_i, S_i, R_i, G_i, j_i, d_i)$ and the discrete controller automaton of Figure 5.

Fig. 5 Discrete controller automaton for the protector CL-PROT-SOLO_i.

Actions:	Input:	e , the environment action (stuttering)
	Output:	$\text{snapshot}(y)_j$, for each valuation y of Y_{VEHICLES}
		$\text{brake}(i)_j$
		$\text{unbrake}(i)_j$
Variables:	Internal:	$\text{send}_j \in \{\text{brake}, \text{unbrake}, \text{null}\}$, initially null
Discrete Transitions:		
<hr/>		
e		$\text{brake}(i)_j$
Eff: None		Pre: $\text{send}_j = \text{brake}$
		Eff: $\text{send}_j := \text{null}$
$\text{snapshot}(y)_j$		$\text{unbrake}(i)_j$
Eff: if $\exists i' \in I, i' \neq i$ such that		Pre: $\text{send}_j = \text{unbrake}$
$y \notin \text{disjoint-claimed-tracks}(i, i', d)$		Eff: $\text{send}_j := \text{null}$
$\wedge (y.x_i < y.x_{i'})$		
then		
$\text{send}_j := \text{brake}$		
else		
$\text{send}_j := \text{unbrake}$		
<hr/>		
Trajectories:		
$w.\text{send}_j \equiv \text{null}$		

Lemma 2. *The protector CL-PROT-SOLO_i guarantees G_i in VEHICLES starting from S_i given R_i .*

Lemma 3. *The protector CL-PROT = $\prod_{i \in I} \text{CL-PROT-SOLO}_i$ for the VEHICLES automaton guarantees $\bigcap_{i \in I} G_i$ in the VEHICLES automaton starting from $\bigcap_{i \in I} S_i$ given $P_{\text{not-overspeed}}$.*

Lemma 3 is shown by combining Lemma 2 and Theorem 3 and realizing that the second condition of Theorem 3 does not hold.

6.3 Example 3: Collision Avoidance on Merging Tracks

In this section, we present a protector, called MERGE-PROT, that guarantees that none of the n vehicles that are traveling on a track involving a Y-shaped merge collide, provided that they are all abiding by the speed limit \dot{c}_{\max} . The MERGE-PROT protector is defined as the composition of $n(n-1)/2$ separate copies of another protector called MERGE-PROT-PAIR _{$\{i, i'\}$} , one copy for each unordered pair of vehicles $\{i, i'\}$, where $i, i' \in I, i \neq i'$. Each of these MERGE-PROT-PAIR _{$\{i, i'\}$} protectors, for $i, i' \in I, i \neq i'$, guarantees that the vehicles i and i' do not collide into each other, provided that all the vehicles are abiding by the speed limit and the vehicles of all other vehicle pairs do not collide between themselves.

We augment the VEHICLES automaton to involve a track topology consisting of a Y-shaped merge. This is done by replacing the position component of a vehicle's state with a location component — a component that specifies the track

on which the vehicle is traveling and the vehicle's position with respect to the merge point — and update the definitions of the discrete steps and the trajectories of the VEHICLES automaton to handle the location variables. Furthermore, we replace the brake and unbrake input actions of the VEHICLES automaton with protect input actions which allow single protectors to instruct sets of vehicles to apply their “emergency” brakes. Finally, we augment the definitions of the discrete actions pertaining to vehicle collisions such that the blame for a particular collision is assigned to either only the trailing vehicle, if one vehicle collides into the other vehicle from behind, or both vehicles, if the vehicles collide sideways while merging. The resulting physical plant automaton is henceforth referred to as MERGE-VEHICLES.

Let $PP_{\{i,i'\}}$ be the MERGE-VEHICLES automaton. Let the port $j_{\{i,i'\}}$ and the sampling period $d_{\{i,i'\}}$ be the port and sampling period with which the protector MERGE-PROT-PAIR $_{\{i,i'\}}$ communicates with the MERGE-VEHICLES automaton. Let $G_{\{i,i'\}}$ be the set of states in which the vehicles i and i' have not collided into each other, i.e., $G_{\{i,i'\}} = \text{VALID} - P_{\text{collided}(i,i')} - P_{\text{collided}(i',i)}$ (Appendix B). Let $R_{\{i,i'\}}$ be the set of states of the MERGE-VEHICLES automaton in which all the vehicles are abiding by the speed limit and in which the vehicles of all other vehicle pairs have not collided into each other, i.e., $R_{\{i,i'\}} = P_{\text{not-overspeed}} \cap \left(\bigcap_{i'',i''' \in I, i'' \neq i''', \{i'',i'''\} \neq \{i,i'\}} G_{\{i'',i'''\}} \right)$ (Appendix B). Finally, let $S_{\{i,i'\}}$ be the set $\text{safe}_{PP_{\{i,i'\}}, R_{\{i,i'\}}, G_{\{i,i'\}}, j_{\{i,i'\}}}$.

We define the protector MERGE-PROT-PAIR $_{\{i,i'\}}$ to be the composition of $\text{Sensor}(PP_{\{i,i'\}}, S_{\{i,i'\}}, R_{\{i,i'\}}, G_{\{i,i'\}}, j_{\{i,i'\}}, d_{\{i,i'\}})$ and a discrete controller automaton whose braking strategy is as follows. The discrete controller automaton is allowed to brake the vehicles i and i' only if the sections of the track they claim in time $d_{\{i,i'\}}$ overlap. Given that the vehicles i and i' are indeed involved in such a claim overlap, there are two possible scenarios depending on whether the locations of the vehicles i and i' are comparable, or not. If their locations are comparable, then the vehicle i is instructed to brake if it trails the vehicle i' ; otherwise, the vehicle i' is instructed to brake. On the other hand, if the vehicle locations are not comparable, the vehicle i is instructed to brake either if *only* the vehicle i' owns the merge point, or if both or neither vehicles own the merge point and the vehicle i is traveling on the left branch of the merge; otherwise, the vehicle i' is instructed to brake. In the latter case, we choose to brake the vehicle traveling on the left branch for no particular reason. In fact, it is plausible to brake either or both of the vehicles involved in the claim overlap.

Lemma 4. *The protector MERGE-PROT-PAIR $_{\{i,i'\}}$ guarantees that the MERGE-VEHICLES automaton remains within $G_{\{i,i'\}}$ starting from $S_{\{i,i'\}}$ given $R_{\{i,i'\}}$.*

Lemma 5. *The protector MERGE-PROT = $\prod_{i,i' \in I, i \neq i'} \text{MERGE-PROT-PAIR}_{\{i,i'\}}$ for the MERGE-VEHICLES automaton guarantees $\bigcap_{i,i' \in I, i \neq i'} G_{\{i,i'\}}$ in MERGE-VEHICLES starting from $\bigcap_{i,i' \in I, i \neq i'} S_{\{i,i'\}}$ given $P_{\text{not-overspeed}}$.*

Lemma 5 is shown by combining Lemma 4 and Theorem 3 and realizing that the second condition of Theorem 3 does not hold.

6.4 Example 4: Collision Avoidance on a General Graph of Tracks

In this section, we present a protector, called GRAPH-PROT, that guarantees that none of the n vehicles traveling on a directed graph of tracks comprised of Y-shaped merges and diverges collide, provided that they are all abiding by the speed limit \dot{c}_{max} . As in Section 6.3, the GRAPH-PROT protector is defined as the composition of $n(n-1)/2$ separate copies of another protector called GRAPH-PROT-PAIR $_{\{i,i'\}}$, one copy for each unordered pair of vehicles $\{i,i'\}$, where $i,i' \in I, i \neq i'$. Each of the GRAPH-PROT-PAIR $_{\{i,i'\}}$ protectors, for $i,i' \in I, i \neq i'$, guarantees that the vehicles i and i' do not collide into each other, provided that all the vehicles are abiding by the speed limit and the vehicles of all other vehicle pairs do not collide between themselves.

We augment the MERGE-VEHICLES automaton to involve a general track topology consisting of a directed graph G of Y-shaped merges and diverges. All the edges of the graph G are assumed to be of sufficient length to rule out collisions among vehicles that are neither on identical, nor on contiguous edges and all cycles of the graph G are assumed to have at least three edges. Moreover, in order to brake the topological symmetry in merge situations, we associate with each edge of the track topology a unique and totally ordered priority index. The resulting physical plant automaton is henceforth referred to as GRAPH-VEHICLES.

Letting $PP_{\{i,i'\}}$, $S_{\{i,i'\}}$, $R_{\{i,i'\}}$, $G_{\{i,i'\}}$, $j_{\{i,i'\}}$, and $d_{\{i,i'\}}$ be as defined in Section 6.3, we define the GRAPH-PROT-PAIR $_{\{i,i'\}}$ automaton to be the composition of $Sensor(PP_{\{i,i'\}}, S_{\{i,i'\}}, R_{\{i,i'\}}, G_{\{i,i'\}}, j_{\{i,i'\}}, d_{\{i,i'\}})$ and a discrete controller automaton whose braking strategy is as follows. The discrete controller automaton is allowed to brake the vehicles i and i' only if the sections of the track they claim in $d_{\{i,i'\}}$ time units overlap. Given that the vehicles i and i' are indeed involved in such a claim overlap, there are two possible scenarios depending on whether the vehicles i and i' are traveling in succession, or on adjacent tracks. If the vehicles are traveling in succession, then the vehicle i is instructed to brake if it trails the vehicle i' ; otherwise, the vehicle i' is instructed to brake. On the other hand, if the vehicles i and i' are traveling on adjacent edges, the vehicle i is instructed to brake either if *only* the vehicle i' owns the merge point, or if both or neither vehicles own the merge point and the vehicle i' is traveling on the edge of greater priority; otherwise, the vehicle i' is instructed to brake.

Lemma 6. *The protector GRAPH-PROT-PAIR $_{\{i,i'\}}$ guarantees that the GRAPH-VEHICLES automaton remains within $G_{\{i,i'\}}$ starting from $S_{\{i,i'\}}$ given $R_{\{i,i'\}}$.*

Lemma 7. *The protector GRAPH-PROT = $\prod_{i,i' \in I, i \neq i'} \text{GRAPH-PROT-PAIR}_{\{i,i'\}}$ for the GRAPH-VEHICLES automaton guarantees $\bigcap_{i,i' \in I, i \neq i'} G_{\{i,i'\}}$ in GRAPH-VEHICLES starting from $\bigcap_{i,i' \in I, i \neq i'} S_{\{i,i'\}}$ given $P_{\text{not-overspeed}}$.*

Lemma 7 is shown by combining Lemma 6 and Theorem 3 and realizing that the second condition of Theorem 3 does not hold.

6.5 Composing the Overspeed and Collision Protectors

In the previous sections, we presented example protectors whose correct operation required that the physical plant automaton at hand satisfied particular

properties. For example, in the case of the VEHICLES automaton of Section 5, the overspeed protector OS-PROT of Section 6.1 assumes that none of the vehicles collide among themselves and the collision protector CL-PROT of Section 6.2 assumes that none of the vehicles exceed the speed limit. Using Theorem 3 it can be shown that the composition $\text{OS-PROT} \times \text{CL-PROT}$ is a protector for the VEHICLES automaton that guarantees that the vehicles in the VEHICLES automaton neither exceed the speed limit, nor collide among themselves. In fact, realizing that the OS-PROT protector extends, virtually unchanged, to the MERGE-VEHICLES and GRAPH-VEHICLES automata, such composition results extend to the MERGE-VEHICLES and GRAPH-VEHICLES automata by composing the OS-PROT protector with the MERGE-PROT and GRAPH-PROT protectors, respectively.

7 Conclusions

In this paper, we demonstrate how formal analysis techniques using the hybrid I/O automaton model can be applied to the specification and verification of hybrid systems whose structure adheres to the protection subsystem paradigm. We propose a parameterized abstract protector model which allows simple specification of an abstract protector for any hybrid system of this form. Such specification is obtained by defining the physical system, the start states, the sets of guarantee and reliance, and the port and sampling period with which the protector communicates with the physical plant. The proof of correctness of the abstract model leads to simple correctness proofs of the protector implementations for particular instantiations of the abstract model. Finally, the composition of independent, and even dependent protectors under mild conditions, guarantees the conjunction of the safety properties guaranteed by the individual protectors. The examples presented in this paper show that the proposed formal framework provides a precise and succinct protector specification, involves simple and straight forward proof methodology, and scales to complex hybrid systems through abstraction and modular decomposition.

Acknowledgments

We would hereby like to thank Dr. Steven L. Spielman of Raytheon Corporation and Norman M. Delisle formerly of Raytheon Corporation for helpful discussions regarding the PRT 2000TM. We are grateful for having the opportunity to develop our formal modeling framework on the basis of a real application. We would also like to thank the submission's reviewers for their helpful suggestions and constructive comments.

References

1. Michael S. Branicky. *Studies in Hybrid Systems: Modeling, Analysis, and Control*. Doctor of Science Thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, June 1995.

2. Ekaterina Dolginova and Nancy A. Lynch. Safety Verification for Automated Platoon Maneuvers: A Case Study. In Oded Maler, editor, *Proc. International Workshop on Hybrid and Real-Time Systems (HART'97)*, volume 1201 of *Lecture Notes in Computer Science*, pages 154–170. Springer-Verlag, 1997. The International Workshop on Hybrid and Real-Time Systems took place in Grenoble, France, in March 1997.
3. Rainer Gawlick, Roberto Segala, Jørgen Søgaaard-Andersen, and Nancy A. Lynch. Liveness in Timed and Untimed Systems. Technical Report MIT/LCS/TR-587, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, December 1993.
4. Rainer Gawlick, Roberto Segala, Jørgen Søgaaard-Andersen, and Nancy A. Lynch. Liveness in Timed and Untimed Systems. In Serge Abiteboul and Eli Shamir, editors, *Proc. 21st International Colloquium on Automata, Languages and Programming (ICALP'94)*, volume 820 of *Lecture Notes in Computer Science*, pages 166–177. Springer-Verlag, 1994. The 21st International Colloquium on Automata, Languages and Programming (ICALP'94) took place in Jerusalem, Israel, in July 1994. Full version appeared as Ref. 3.
5. Robert L. Grossman, Anil Nerode, Anders P. Ravn, and Hans Rischel, editors. *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993. This volume of LNCS was inspired by a workshop on the Theory of Hybrid Systems, held on Oct. 19–21, 1992 at the Technical University, Lyngby, Denmark, and by a prior Hybrid Systems Workshop, held on June 10–12, 1991 at the Mathematical Sciences Institute, Cornell University.
6. Carolos Livadas. Formal Verification of Safety-Critical Hybrid Systems. Master of Engineering Thesis, Dept. of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, September 1997.
7. John Lygeros. *Hierarchical, Hybrid Control of Large Scale Systems*. Doctor of Philosophy Thesis, Dept. of Electrical Engineering and Computer Sciences, University of California, Berkeley, May 1996.
8. John Lygeros, Datta N. Godbole, and Shankar Sastry. A Verified Hybrid Controller for Automated Vehicles. In *35th IEEE Conference on Decision and Control (CDC'96)*, pages 2289–2294, Kobe, Japan, December 1996.
9. John Lygeros and Nancy Lynch. On the Formal Verification of the TCAS Conflict Resolution Algorithm. In *36th IEEE Conference on Decision and Control (CDC'97)*, San Diego, CA, December 1997. To appear.
10. Nancy Lynch, Roberto Segala, Frits Vaandrager, and H. B. Weinberg. Hybrid I/O Automata. Technical Memo MIT/LCS/TM-544, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, December 1995.
11. Nancy Lynch, Roberto Segala, Frits Vaandrager, and H. B. Weinberg. Hybrid I/O Automata. In R. Alur, T. Henzinger, and E. Sontag, editors, *Proc. DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems, Hybrid Systems III: Verification and Control*, volume 1066 of *Lecture Notes in Computer Science*, pages 496–510. Springer-Verlag, 1996. The DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems took place in New Brunswick, New Jersey, in October 1995.
12. Nancy Lynch, Roberto Segala, Frits Vaandrager, and H. B. Weinberg. Hybrid I/O Automata. Preprint/Work in Progress. Preliminary versions appeared as Refs. 10 and 11, June 1997.
13. Nancy Lynch and Frits Vaandrager. Forward and Backward Simulations — Part II: Timing-Based Systems. Technical Memo MIT/LCS/TM-487.c, Labora-

- tory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, April 1995.
14. Nancy Lynch and Frits Vaandrager. Forward and Backward Simulations — Part II: Timing-Based Systems. *Information and Computation*, 128(1):1–25, July 1996. Preliminary version appeared as Ref. 13.
 15. George J. Pappas, Claire Tomlin, and Shankar Sastry. Conflict Resolution in Multi-Agent Hybrid Systems. In *35th IEEE Conference on Decision and Control (CDC'96)*, Kobe, Japan, December 1996.
 16. Amir Pnueli and Joseph Sifakis, editors. *Special Issue on Hybrid Systems*, volume 138, part 1 of *Theoretical Computer Science*. Elsevier Science Publishers, February 1995.
 17. H. B. Weinberg, Nancy Lynch, and Norman Delisle. Verification of Automated Vehicle Protection Systems. In R. Alur, T. Henzinger, and E. Sontag, editors, *Hybrid Systems III: Verification and Control*, volume 1066 of *Lecture Notes in Computer Science*, pages 101–113. Springer-Verlag, 1996.

A Derived Variables and Sets of the VEHICLES Automaton

$E_i \in \mathcal{P}(\mathbb{R})$, defined by $E_i = [x_i, x_i + c_{len}]$.

$collided(i, *) \in \text{Bool}$, for $i \in I$, defined by $collided(i, *) = \bigvee_{i' \in I, i' \neq i} collided(i, i')$.

$collided(*, i) \in \text{Bool}$, for $i \in I$, defined by $collided(*, i) = \bigvee_{i' \in I, i' \neq i} collided(i', i)$.

$collided(*, i, *) \in \text{Bool}$, for $i \in I$, defined by $collided(*, i, *) = collided(*, i) \vee collided(i, *)$.

$VALID \subseteq \text{states}(\text{VEHICLES})$, defined by

$$\begin{aligned}
 VALID = \{ & p \in \text{states}(\text{VEHICLES}) \mid \\
 & 1. \nexists i, i' \in I, i \neq i' \text{ such that the set } p.E_i \cap p.E_{i'} \text{ is a positive length} \\
 & \quad \text{closed interval of } \mathbb{R}. \\
 & 2. p.\dot{x}_i \geq 0, \text{ for all } i \in I. \\
 & 3. \text{ If } \neg p.collided(*, i, *) \text{ then } p.\ddot{x}_i \in [\ddot{c}_{min}, \ddot{c}_{max}], \text{ for all } i \in I. \\
 & 4. \text{ If } \neg p.collided(*, i, *) \wedge p.brake(i) \text{ then if } p.\dot{x}_i = 0 \text{ then } p.\ddot{x}_i = 0 \\
 & \quad \text{else } p.\ddot{x}_i = \ddot{c}_{brake}, \text{ for all } i \in I. \}
 \end{aligned}$$

$stop-dist_i \in \mathbb{R}^{\geq 0}$, for all $i \in I$, defined by

$$stop-dist_i = -\frac{\dot{x}_i^2}{2\ddot{c}_{brake}}$$

$max-range_i(t) \in \mathbb{R}^{\geq 0}$, for all $i \in I$ and $t \in \mathbb{R}^{\geq 0}$, defined by

$$max-range_i(t) = \begin{cases} \dot{x}_i \Delta t + \frac{1}{2} \ddot{c}_{max} \Delta t^2 + \dot{c}_{max}(t - \Delta t), & \text{if } \dot{x}_i \leq \dot{c}_{max}, \text{ and} \\ \quad \text{where } \Delta t = \min\left(t, \frac{\dot{c}_{max} - \dot{x}_i}{\ddot{c}_{max}}\right) & \\ \dot{x}_i \Delta t + \frac{1}{2} \ddot{c}_{brake} \Delta t^2 + \dot{c}_{max}(t - \Delta t), & \text{otherwise.} \\ \quad \text{where } \Delta t = \min\left(t, \frac{\dot{c}_{max} - \dot{x}_i}{\ddot{c}_{brake}}\right) & \end{cases}$$

$max-vel_i(t) \in \mathbb{R}^{\geq 0}$, for all $i \in I$ and $t \in \mathbb{R}^{\geq 0}$, defined by

$$max-vel_i(t) = \begin{cases} \min(\dot{c}_{max}, \dot{x}_i + t\ddot{c}_{max}) & \text{if } \dot{x}_i \leq \dot{c}_{max}, \text{ and} \\ \max(\dot{c}_{max}, \dot{x}_i + t\ddot{c}_{brake}) & \text{otherwise.} \end{cases}$$

$O_i \subseteq \mathbb{R}$, for all $i \in I$, defined by

$$O_i = [x_i, x_i + \text{stop-dist}_i + c_{len}]$$

$C_i(t) \subseteq \mathbb{R}$, for all $i \in I$ and $t \in \mathbb{R}^{\geq 0}$, defined by

$$C_i(t) = [x_i, x_i + \text{max-range}_i(t) - \text{max-vel}_i(t)^2 / (2\ddot{c}_{brake}) + c_{len}]$$

B Auxiliary Sets for the VEHICLES Automaton

$P_{\text{overspeed}}(i) \subseteq \text{VALID}$, for $i \in I$, defined by

$$P_{\text{overspeed}}(i) = \{p \in \text{VALID} \mid p.\dot{x}_i > \dot{c}_{max}\}$$

$P_{\text{overspeed}} \subseteq \text{VALID}$, defined by $P_{\text{overspeed}} = \bigcup_{i \in I} P_{\text{overspeed}}(i)$.

$P_{\text{not-overspeed}} \subseteq \text{VALID}$, defined by $P_{\text{not-overspeed}} = \text{VALID} - P_{\text{overspeed}}$.

$P_{\text{collided}}(i, i') \subseteq \text{VALID}$, for $i, i' \in I, i \neq i'$, defined by

$$P_{\text{collided}}(i, i') = \{p \in \text{VALID} \mid p.\text{collided}(i, i') = \text{True}\}$$

$P_{\text{collided}}(i) \subseteq \text{VALID}$, defined by $P_{\text{collided}}(i) = \bigcup_{i' \in I, i' \neq i} P_{\text{collided}}(i, i')$.

$P_{\text{collided}} \subseteq \text{VALID}$, defined by $P_{\text{collided}} = \bigcup_{i \in I} P_{\text{collided}}(i) = \bigcup_{i, i' \in I, i \neq i'} P_{\text{collided}}(i, i')$.

$P_{\text{not-collided}} \subseteq \text{VALID}$, defined by $P_{\text{not-collided}} = \text{VALID} - P_{\text{collided}}$.

$\text{disjoint-extents}(i, i') \subseteq \text{VALID}$, for $i, i' \in I, i \neq i'$, defined by

$$\text{disjoint-extents}(i, i') = \{p \in \text{VALID} \mid p.E_i \cap p.E_{i'} = \emptyset\}$$

$P_E \subseteq \text{VALID}$, defined by

$$P_E = \bigcap_{i, i' \in I, i \neq i'} \text{disjoint-extents}(i, i')$$

$\text{disjoint-owned-tracks}(i, i') \subseteq \text{VALID}$, for $i, i' \in I, i \neq i'$, defined by

$$\text{disjoint-owned-tracks}(i, i') = \{p \in \text{VALID} \mid p.O_i \cap p.O_{i'} = \emptyset\}$$

$P_O \subseteq \text{VALID}$, defined by

$$P_O = \bigcap_{i, i' \in I, i \neq i'} \text{disjoint-owned-tracks}(i, i')$$

$\text{disjoint-claimed-tracks}(i, i', t) \subseteq \text{VALID}$, for $i, i' \in I, i \neq i'$, and $t \in \mathbb{R}^{\geq 0}$, defined by

$$\text{disjoint-claimed-tracks}(i, i', t) = \{p \in \text{VALID} \mid p.C_i(t) \cap p.C_{i'}(t) = \emptyset\}$$

$P_{C(t)} \subseteq \text{VALID}$, for $t \in \mathbb{R}^{\geq 0}$, defined by

$$P_{C(t)} = \bigcap_{i, i' \in I, i \neq i'} \text{disjoint-claimed-tracks}(i, i', t)$$

$P_{B_{i,j}} \subseteq \text{VALID}$, defined by

$$P_{B_{i,j}} = \{p \in \text{VALID} \mid p.\text{brake-req}(i, j) = \text{True}\}$$

Strings of Vehicles: Modeling and Safety Conditions*

John Lygeros[†] and Nancy Lynch[‡]

[†]Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
Berkeley, CA 94720-1770
lygeros@eecs.berkeley.edu

[‡]Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
lynch@lcs.mit.edu

Abstract. Motivated by our work on Automated Highway Systems (AHS), we consider a physical system, the *string of vehicles* and construct a natural model for it in the Hybrid Input/Output Automaton formalism. We describe a special maneuver that may have to be executed by the system, the *emergency deceleration maneuver*, and derive necessary and sufficient conditions on the system parameters under which this maneuver can be executed in safety. We conclude by giving a brief discussion of the implications of our results for the design of an AHS that allows the formation of platoons of vehicles.

1 Introduction

Hybrid systems have attracted the attention of both computer theorists and control engineers. Our work ultimately aims at a rapprochement of these two perspectives. Here we use a combination of techniques from the two areas to address a specific problem in transportation. This is the problem of the safety of a collection of vehicles traveling one behind the other in a single lane; we refer to such a collection as a *string of vehicles*. The problem is hybrid as it involves both continuous vehicle motion and (possibly) collisions, which in our setting are treated as discrete velocity changes. We try to establish conditions under which a string of vehicles will be safe while executing a particular maneuver.

We start by developing a detailed model for the system in the *Hybrid Input/Output Automaton* modeling framework (Section 2). Modest extensions of the original framework of [1] are needed to capture all the phenomena of interest for this problem. Then, in Section 3 we introduce the emergency deceleration

* Research supported by California PATH under MOU-238 and MOU 288, ARPA contract F19628-95-C-0118, AFOSR contract F49620-97-1-0337, AFOSR contract F49620-94-1-0199 and the U.S. Department of Transportation DTRS95G-0001.

maneuver, whose safety analysis is the primary focus of this paper. We give some necessary and some sufficient conditions under which the safety of the maneuver can be guaranteed. Finally, in Section 4, we discuss the implications of our results in the context of platooning of vehicles.

We believe our work is potentially of both theoretical and practical importance. On the theoretical side we hope that the results presented here will be extended to a general methodology for dealing with hybrid systems, one where continuous and discrete techniques are combined in a coherent framework. The practical implications of our work are more immediate. Our results indicate that the design of specialized emergency maneuvers may be crucial to the success of an automated highway system that allows for the formation of platoons.

2 Vehicle String Model

2.1 Overview of the Modeling Formalism

Based on the work of [1], we consider a hybrid automaton, A , as a dynamical system that describes the evolution of a finite collection of variables, V_A . Variables are typed; for each $v \in V_A$ let $type(v)$ denote the type of v . For each $Z \subseteq V_A$, a *valuation* of Z is a function that to each $v \in Z$ assigns a value in $type(v)$. Let \mathbf{Z} denote the set of valuations of Z ; we refer to $s \in \mathbf{V}_A$ as a *system state*. In this paper we assume that the evolution of the variables is over the set $T^{\geq 0} = \{t \in \mathbb{R} | t \geq 0\}$. The evolution of the variables involves both continuous and discrete dynamics. Continuous dynamics are encoded in terms of *trajectories* over V_A , that is functions that map intervals of $T^{\geq 0}$ to \mathbf{V}_A . Discrete dynamics are encoded by *actions*. Upon the occurrence of an action the system state instantaneously “jumps” to a new value. We use Σ_A to denote the set of actions that affect the evolution of A .

More formally, a *hybrid automaton*, A is a collection $(U_A, X_A, Y_A, \Sigma_A^{in}, \Sigma_A^{int}, \Sigma_A^{out}, \Theta_A, \mathcal{D}_A, \mathcal{W}_A)$ consisting of:

- Three disjoint sets U_A , X_A , and Y_A of variables, called *input*, *internal*, and *output variables*, respectively. We set $V_A = U_A \cup X_A \cup Y_A$.
- Three disjoint sets Σ_A^{in} , Σ_A^{int} , and Σ_A^{out} of actions, called *input*, *internal*, and *output actions*, respectively. We set $\Sigma_A = \Sigma_A^{in} \cup \Sigma_A^{int} \cup \Sigma_A^{out}$.
- A non-empty set $\Theta_A \subseteq \mathbf{V}_A$ of *initial states*.
- A set $\mathcal{D}_A \subseteq \mathbf{V}_A \times \Sigma_A \times \mathbf{V}_A$ of *discrete transitions*.
- A set \mathcal{W}_A of *trajectories* over V_A .

Some technical axioms are imposed on the above sets to guarantee that the definitions are consistent. The axioms introduced in [1] are too restrictive for the application considered here; fortunately the extensions needed are fairly straightforward.

An *execution*, α , of A is an alternating sequence $\alpha = w_0 a_1 w_1 a_2 w_2 \dots$, finite or infinite, where for all i , $a_i \in \Sigma_A$, $w_i \in \mathcal{W}_A$ defined over a left closed time interval and $fstate(w_0) \in \Theta_A$. In addition, if α is a finite sequence then it ends

with a trajectory and if w_i is not the last trajectory its domain is right-closed and $(lstate(w_i), a_{i+1}, fstate(w_{i+1})) \in \mathcal{D}_A$. Here $fstate(w)$ and $lstate(w)$ denote the initial and final states of a trajectory w . An execution is called *finite* if it is a finite sequence and the domain of its final trajectory is a right-closed interval. A state $s \in \mathbf{V}_A$ is called *reachable* if it is the last state of a finite execution.

Hybrid automata "communicate" through shared variables and shared actions. Consider two automata A and B with $X_A \cap V_B = X_B \cap V_A = Y_B \cap Y_A = \emptyset$ and $\Sigma_B^{int} \cap \Sigma_A = \Sigma_A^{int} \cap \Sigma_B = \Sigma_A^{out} \cap \Sigma_B^{out} = \emptyset$. Under some mild technical assumptions, the *composition*, $A \times B$, of A and B can be defined as a new hybrid automaton with $U_{A \times B} = (U_A \cup U_B) \setminus (Y_A \cup Y_B)$, $X_{A \times B} = X_A \cup X_B$, $Y_{A \times B} = Y_A \cup Y_B$ and similarly for the actions. $\Theta_{A \times B}$, $\mathcal{D}_{A \times B}$ and $\mathcal{W}_{A \times B}$ are such that the executions of $A \times B$ are also executions of each automaton when restricted to the corresponding variables and actions.

A *derived variable* of A is a function on \mathbf{V}_A . Derived variables will be used to simplify the system description, but also to facilitate the analysis. A *property* of A is a boolean derived variable. A property is *stable* if whenever it is true at some state it is also true at all states reachable from that state. A property is *invariant* if it is true at all reachable states. Typically properties will be shown to be stable or invariant by an induction argument on the length of an execution. It is easy to show that:

Lemma 1 *Assume that for all reachable states s of A , P true at s implies P true at s' for all s' such that either there exists $w \in \mathcal{W}_A$ with right closed domain and $fstate(w) = s$ and $lstate(w) = s'$, or, there exists $a \in \Sigma_A$ with $(s, a, s') \in \mathcal{D}_A$. Then P is a stable property. If further P is true at all $s \in \Theta_A$, then P is an invariant property.*

In some places differential equations will be used to simplify the description of the set \mathcal{W}_A . In such cases \mathcal{W}_A is assumed to be populated by all trajectories generated by the differential equation in the usual way. To simplify the description of \mathcal{D}_A , we will assign a *precondition* and an *effect* to each action. The precondition is a predicate on \mathbf{V}_A while the effect is a predicate on $\mathbf{V}_A \times \mathbf{V}_A$. The action can take place only from states that satisfy the precondition; moreover, the states before and after the transition should be such that the effect is satisfied. When no confusion can arise we use v' to denote the value of variable v after an action.

2.2 String Model

Consider a string of N vehicles (Figure 1) moving one behind the other in a single lane, with vehicle 0 coming first. The overall model will be the composition of a number of automata (Figure 2). The *plant* will be a hybrid automaton containing the dynamics of all the vehicles in the string. Each vehicle is equipped with *sensors* and *controllers*. The sensor automaton S_i reads the values of the plant output variables as inputs and produces real valued output variables. The controller automaton, C_i , reads the corresponding sensor output variables and

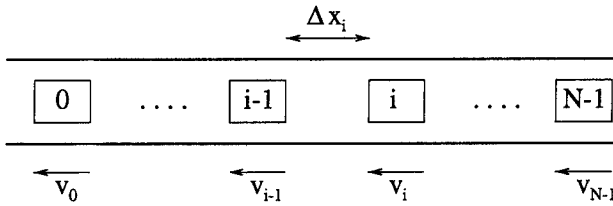


Fig. 1. A string of vehicles

uses them to generate the input variables of the plant. The S_i and C_i may have internal variables and actions. In this paper we assume that the sensor and controller automata are simple input/output maps and concentrate on the development of a realistic plant model.

The plant is modeled by an automaton $P = (U_P, X_P, Y_P, \Sigma_P^{in}, \Sigma_P^{int}, \Sigma_P^{out}, \Theta_P, \mathcal{D}_P, \mathcal{W}_P)$. P has no input and no output actions, hence $\Sigma_P^{in} = \Sigma_P^{out} = \emptyset$. Here we are only interested in answering questions of "safety", encoded in terms of possible collisions among the vehicles of the string. The answers to these questions will depend on the relative spacing and the velocities of the vehicles, but not their absolute position on the road. Let Δx_i denote the spacing between vehicle i and $i - 1$, v_i the speed of vehicle i , acc_i its acceleration and u_i its commanded acceleration² and define $x_i = [\Delta x_i \ v_i] \in \mathbb{R}^2$, $x = [x_0 \ \dots \ x_{N-1}] \in \mathbb{R}^{2N}$ and $u = [u_0 \ \dots \ u_{N-1}] \in \mathbb{R}^N$. Also consider a collection of boolean variables $Touching = \{Touching_1, \dots, Touching_{N-1}\}$; the evolution of these variables (Section 2.2) will be such that $Touching_i$ is true whenever vehicle i is touching vehicle $i - 1$. Define the internal and input variables as $X_P = \{x, acc, Touching\}$ and $U_P = \{u\}$ respectively. Physical limitations constrain the valuations of the input variables to lie in a rectangular compact set, i.e. $u_i(t) \in [a_i^{min}, a_i^{max}]$ for all i and for all t . The values of a_i^{min} and a_i^{max} are determined by the vehicle characteristics (engine, brakes, tires, etc.). To ensure that the model is realistic we impose the following assumption on Θ_P and the input constraints.

Assumption 1 For all i , $\Delta x_i(0) \geq 0$, $v_i(0) \geq 0$, $Touching_i(0) = \text{False}$ and $a_i^{min} < 0 < a_i^{max}$

Discrete Dynamics The continuous system evolution can be interrupted by three classes of internal actions: collisions, vehicles touching with zero relative velocity (and subsequently "pushing" against one another) and vehicles moving apart (after having touched). We assume that the continuous evolution stops as soon as the precondition of an action becomes true, to allow the action to take place. All variables not explicitly mentioned in the effect are assumed to be unaffected by the action.

² As discussed in Section 2.2, the commanded and actual acceleration may differ when vehicles are touching and pushing each other.

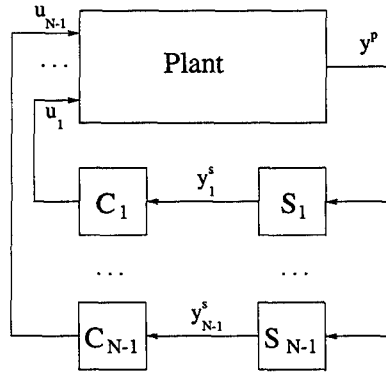


Fig. 2. System modules

Consider first the case of collisions. Let *Collision_i* be an internal action that takes place whenever vehicle i collides with vehicle $i - 1$. The precondition for *Collision_i* is:

$$(\Delta x_i = 0) \wedge (v_i > v_{i-1}) \quad (1)$$

To determine the effect of the action we use a simple collision model. To determine v_i and v_{i-1} after the collision we solve a pair of equations:

$$M_i v'_i + M_{i-1} v'_{i-1} = M_i v_i + M_{i-1} v_{i-1} \quad (2)$$

$$v'_{i-1} - v'_i = (v_i - v_{i-1}) \alpha_i \quad (3)$$

where M_i is the mass of vehicle i and α_i is the coefficient of restitution, a measure of the energy lost in the collision. Equation (2) is the *conservation of momentum equation* while Equation (3) is referred to as the *restitution equation*. By appropriate choice of α (possibly as a function of the speeds) this collision model can capture a wide range of collision scenarios. To maintain a certain level of generality in the subsequent discussion we will typically assume that the coefficient of restitution is a function of the relative velocity $v_{i-1} - v_i$ at impact and will denote it by $\alpha_i(\cdot)$. To ensure that the model is realistic we impose the following assumption:

Assumption 2 For all i , $M_i > 0$ and $\alpha_i(v) \in [0, 1]$ for all $v > 0$.

Note that in general vehicles may end up going backwards as a result of collisions if, for example, a light vehicle elastically hits a slowly moving heavy vehicle (i.e. $M_i \ll M_{i-1}$, $\alpha_i \approx 1$ and $v_{i-1} \approx 0$).

Multiple instantaneous collisions are also possible in this setting. These are situations where there exist N_1 and N_2 with $0 \leq N_1 < N_2 < N$ such that $\Delta x_{N_1} \neq 0$, $\Delta x_{N_2+1} \neq 0$ (if any) and for all i with $N_1 < i \leq N_2$, $\Delta x_i = 0$ and $v_i > v_{i-1}$. The value, x' , of the state after the collision again satisfies $\Delta x'_i = \Delta x_i$ for all i and $v'_i = v_i$ for all $i < N_1$ or $i > N_2$. To determine the values of v_i for $N_1 \leq i \leq N_2$ we resolve the multiple collision as a sequence of

pairwise collisions, according to equations (2) and (3). The pairwise resolutions will keep taking place as long as there exists a j with $N_1 < j \leq N_2$ such that $v_j > v_{j-1}$. When this condition is violated we will say that the multiple collision has been resolved. It turns out that, if the masses of the vehicles are unequal or the restitution coefficients α_i are not identically equal to 1, one can construct scenarios where the velocities of the vehicles after the multiple collision has been resolved depend on the order in which the pairwise resolutions were executed. To circumvent this problem we state our theorems and proofs in a way that the results hold for *all possible* orderings of the pairwise resolutions.

Next, let $Touch_i$ be an internal action that takes place whenever vehicle i touches vehicle $i - 1$ with zero relative velocity. The precondition for $Touch_i$ is:

$$(Touching_i = \text{False}) \wedge (\Delta x_i = 0) \wedge (v_i = v_{i-1}) \wedge (acc_i \geq acc_{i-1}) \quad (4)$$

The effect of $Touch_i$ is to declare the two vehicles as touching, i.e. $Touching'_i = \text{True}$.

Finally, consider what happens when vehicles that are touching start moving away from one another. Let $Separate_i$ be an internal action that takes place whenever vehicle i is already touching vehicle $i - 1$ and starts to move away. The precondition for $Separate_i$ is:

$$(Touching_i = \text{True}) \wedge [(acc_i < acc_{i-1}) \vee (v_i < v_{i-1})] \quad (5)$$

The effect of $Separate_i$ is to declare the two vehicles as no longer touching, i.e. $Touching'_i = \text{False}$.

Continuous Dynamics The set of trajectories \mathcal{W}_P will be generated by a dynamical system. Assume there are no vehicles ahead of the string and set $\Delta x_0 \equiv \infty$. Then, for $i = 1, \dots, N - 1$ the laws of motion imply that:

$$\begin{aligned} \Delta x_i(t) &= v_{i-1}(t) - v_i(t) \\ \dot{v}_i(t) &= acc_i(t) \end{aligned}$$

The value of the actual acceleration, acc_i , of vehicle i depends on the acceleration commanded by the controller of that vehicle, u_i , and on whether the vehicle is touching vehicle $i - 1$ or vehicle $i + 1$. In the case when the vehicles are not touching we simply set the actual acceleration equal to the commanded acceleration. The case where vehicles are touching is more complicated. The reason is that when vehicles are pushing against one another, there are forces exerted from one vehicle to the other. Therefore, the actual acceleration of a vehicle depends not only on the acceleration commanded by its own controller, but also on the accelerations commanded by the controllers of the neighboring vehicles that are pushing against it.

To resolve this issue we first introduce some abstract definitions. Consider a nonempty finite subset of the natural numbers $S \subset \mathbb{N}$. S is a *segment* if it consists of consecutive numbers. A *subsegment* of a segment S is any subset of S that is also a segment. For segments S_1 and S_2 with $\min(S_2) = \max(S_1) + 1$

we define their *concatenation* (denoted by $S_1 S_2$) as the segment $S_1 \cup S_2$. A *weighted average function on S* is any function $a : 2^S \rightarrow \mathbb{R}$ such that for all L, R subsegments of S :

$$\min\{a(L), a(R)\} \leq a(LR) \leq \max\{a(L), a(R)\} \quad (6)$$

whenever the concatenation LR is defined. A segment S with a weighted average function a is *unsplittable* if:

$$S = LR \Rightarrow a(L) \leq a(R)$$

A *partition of S* is a finite collection S_1, \dots, S_n where $S = \bigcup_{k=1}^n S_k$ and for all k , S_k is a segment and $S_k \cap S_l = \emptyset$ for $l \neq k$. Without loss of generality assume that $\min(S) = \min(S_1)$ and for all $1 < k \leq n$, $\min(S_k) = \max(S_{k-1}) + 1$ and write $S = S_1 S_2 \dots S_n$. A partition of $S_1 \dots S_n$ of S is called a *maximal partition* if for all $k = 1, \dots, n$, S_k is unsplittable and either $n = 1$ or for all $k = 2, \dots, n$, $a(S_{k-1}) > a(S_k)$.

Theorem 1 *For every segment, S , and every weighted average function, a , on S there exists a unique maximal partition.*

Though interesting, the proof of Theorem 1 is omitted here as it is not necessary for the safety results. An algorithm to construct the maximal partition has also been developed.

Intuitively (returning to the vehicle example) a maximal partition is such that vehicles in an element of the partition are pushing against one another while vehicles in different elements of the partition are moving away from one another. Assume there exist i, j with $0 < i < j < N$ such that vehicles i to j are touching each other. Define the segment $S = \{i, \dots, j\}$ and for every subset $S' \subseteq S$ consider the function:

$$a(S') = \frac{\sum_{k \in S'} M_k u_k}{\sum_{k \in S'} M_k} \quad (7)$$

One can show that a is a weighted average function on S . To determine the acceleration of the vehicles in this collection at a given instant, let $S_1 \dots S_n$ be the maximal partition of S at that instant and for all $k = 1, \dots, n$ set:

$$acc_l = a(S_k) \text{ for all } l \in S_k \quad (8)$$

If one assumes that the force exerted on a vehicle by the road depends only on the commanded acceleration of that vehicle (and not on whether the vehicle is touching other vehicles), then this choice is what one would expect from physical intuition. The total force commanded by all the vehicles determines the acceleration of their combined mass.

2.3 Output Evolution

The output evolution is determined as a function of the evolution of the inputs and states. We assume that in principle all the internal variables can be made available to the controllers. Limitations imposed by current sensing and communication technology should be incorporated in the sensor automata. Therefore the information made available by vehicle i is $y_i^p(t) = [x_i(t) \text{ acc}_i(t)]$. Let $y^p = [y_0^p \dots y_{N-1}^p] \in \mathbb{R}^{3N}$ and define the output variables as $Y_P = \{y^p\}$.

2.4 Model Consistency & Safety Requirements

The following lemma suggests the proposed plant model agrees with basic physical intuition:

Lemma 2 *The plant automaton is such that:*

1. *If E and E' are the kinetic energy before and after Collision $_i$, then $E' \leq E$.*
2. *$\bigwedge_{i=0}^{N-1} [\Delta x_i \geq 0]$ is an invariant property of the plant.*
3. *$\bigwedge_{i=1}^{N-1} [(\text{Touching}_i = \text{True}) \Rightarrow (\Delta x_i = 0)]$ is an invariant property of the plant.*

The kinetic energy of the string is defined as:

$$E = \sum_{i=0}^{N-1} \frac{1}{2} M_i v_i^2$$

The first property shows that (as expected) no energy is generated as a result of the collisions. The second property shows that the model does not allow vehicles to run over one another (a physical impossibility). The last property shows that vehicles are declared as touching by the model only when they are physically touching.

We are interested in defining the system performance in terms of the severity of the collisions experienced by the vehicles. Following [2], we assume that a collision is safe if the relative velocity at impact is below a certain threshold, v_A . A commonly cited threshold is $v_A = 3ms^{-1}$ [2].

Definition 1 *A string is safe if $\bigwedge_{i=1}^{N-1} [(\Delta x_i = 0) \Rightarrow (v_i \leq v_{i-1} + v_A)]$ is an invariant property.*

The main limitation of our model is that it does not account for the lateral motion of the vehicles. We assume that all vehicles effectively move along a straight line. This assumption may be unrealistic, especially in the presence of collisions when large forces and moments can be exerted from one vehicle to another. The situation will be even worse when the vehicles move along a curved road.

3 Safety Conditions for Emergency Deceleration

3.1 Background

The *emergency deceleration maneuver* is a situation where the first vehicle in the string applies maximum deceleration until it comes to a stop, thus endangering the remaining vehicles of the string. It is assumed that the emergency deceleration of vehicle 0 is caused by some abnormal condition, such as a mechanical malfunction or an obstacle. We would like to determine the conditions under which the remaining vehicles can maintain their safety despite this "malicious" behavior of the leader.

The safety of general strings of vehicles has been analyzed using a number of techniques. Most results in the literature start by partly characterizing the string by determining "automata" for the sensors and controllers and then trying to establish the range of initial conditions and parameters for which the string is safe. This type of analysis has led to conditions under which pairs of vehicles are guaranteed not to collide [3, 4] or experience safe collisions [4, 5]. In some cases the conditions have also been extended to longer or even infinite strings [6, 7].

Perhaps the most challenging problem in this area has been the design of controllers for platoons of vehicles. A *platoon* is a string of very tightly spaced vehicles. Typically intra-platoon spacings are of the order of 1-2 meters. The safety of the intra-platoon controllers [6] relies on the assumption that the behavior of the first vehicle is in some sense "reasonable". This means that the controller C_0 takes into account the limitations of the rest of the vehicles in the string when calculating u_0 . This requirement is clearly violated in the case of the emergency deceleration maneuver. It is conjectured however that the platoon is going to be safe even in this case [8]. The justification is that collisions are going to take place in rapid succession, because the vehicles are all close to one another. Therefore, if the speeds of all vehicles are initially the same, the relative velocity at the time of collision is going to be small. We attempt to establish conditions under which this conjecture is true.

The safety of the string under an emergency deceleration maneuver depends on the response of the remaining vehicles of the string to the deceleration of the leader. Here we consider a very simple *default deceleration strategy*. Assume that at time $t = 0$ the leading vehicle applies maximum deceleration, a_0^{min} , until it stops at which point its commanded acceleration becomes 0. After a delay d_i vehicle i also applies a_i^{min} until it comes to a stop. This scenario can be easily encoded in the model discussed above by simple sensor and controller automata. The results discussed here refer to the case $d_i = 0$; some of them directly extend to the more general case.

3.2 Safety Conditions For Strings of Length $N=2$

We first develop conditions for a string of two vehicles to be safe under the default deceleration strategy. These conditions will form the basis of safety results for longer strings. We refer to a two vehicle string as a *pair*. One can easily show that:

Proposition 1 ($v_0 \geq 0$) and ($v_1 \leq 0$) are stable properties for a pair. If ($v_1 \leq 0$) the pair is safe (in particular Collision_1 cannot occur).

To derive more meaningful safety properties consider the derived variables:

$$C_1(\Delta x_1, v_1, v_0) = (a_1^{\min} + a_0^{\min})v_0^2 - 2a_0^{\min}v_0v_1 - 2(a_0^{\min})^2\Delta x_1 \quad (9)$$

$$C_2(\Delta x_1, v_0, v_1) = \frac{a_1^{\min}}{a_0^{\min}}v_0 - v_1 \quad (10)$$

$$P_1(\Delta x_1, v_0, v_1) = (v_0 - v_1)^2 - 2(a_0^{\min} - a_1^{\min})\Delta x_1 - v_A^2 \quad (11)$$

$$P_2(\Delta x_1, v_0, v_1) = v_1^2 - \frac{a_1^{\min}}{a_0^{\min}}v_0^2 + 2a_1^{\min}\Delta x_1 - v_A^2 \quad (12)$$

To simplify the notation we will explicitly mention the function arguments only when necessary. We also introduce a derived boolean variable C given by the expression:

$$C = [(C_1 \leq 0) \wedge (a_0^{\min} \leq a_1^{\min})] \vee [(C_2 \leq 0) \wedge (a_0^{\min} \geq a_1^{\min})] \vee [(v_0 = 0)] \quad (13)$$

P_1, P_2 and C are used to construct safety invariants. A collision can take place either while both vehicles are moving or while vehicle 1 is moving and vehicle 0 has stopped (by Proposition 1 collisions cannot take place once vehicle 1 stops). The property ($P_1 \leq 0$) will encode conditions that guarantee safety if a collision takes place while both vehicles are still moving. ($P_2 \leq 0$) will encode conditions that guarantee that either no collision takes place or a safe collision takes place after vehicle 0 has stopped. The predicate C will be used to distinguish the two cases.

Lemma 3 $(P_1 \leq 0) \vee (v_1 \leq 0)$ is a stable property of the pair.

Proof. $(P_1 \leq 0) \vee (v_1 \leq 0)$ is preserved by Touch_1 and Separate_1 , as both these actions leave $\Delta x_1, v_0$ and v_1 unaffected. Assume $(P_1 \leq 0) \vee (v_1 \leq 0)$ is true when Collision_1 occurs. By Proposition 1 ($v_1 \leq 0$) can not be true in this case. Therefore $(P_1 \leq 0)$ is true, i.e. $P_1(\Delta x_1, v_0, v_1) = P_1(0, v_0, v_1) \leq 0$. Hence, by the restitution equation (3), $(v'_0 - v'_1)^2 = (v_0 - v_1)^2 \alpha_1^2 \leq (v_0 - v_1)^2 \leq v_A^2$, as $\alpha_1 \in [0, 1]$ by Assumption 2. Therefore, $P_1(\Delta x'_1, v'_0, v'_1) = P_1(0, v'_0, v'_1) \leq 0$ and $(P_1 \leq 0) \vee (v_1 \leq 0)$ is again true after Collision_1 .

Assume at some state, s , $(P_1 \leq 0) \vee (v_1 \leq 0)$ is true and consider all trajectories that start at s . If $(v_1 \leq 0)$ is true at s it will also be true at the last state of the trajectory by Proposition 1. If $(P_1 \leq 0) \wedge (v_1 > 0)$ is true at s , consider the variation of P_1 along a trajectory:

$$\begin{aligned} \frac{d}{dt}P_1 &= 2(v_0 - v_1)(acc_0 - acc_1) - 2(a_0^{\min} - a_1^{\min})(v_0 - v_1) \\ &= \begin{cases} 0 & \text{if } (v_0 > 0) \wedge (v_1 > 0) \wedge \neg \text{Touching}_1 \\ 2a_0^{\min}v_1 & \text{if } (v_0 = 0) \wedge (v_1 > 0) \wedge \neg \text{Touching}_1 \\ -2(a_0^{\min} - a_1^{\min})(v_0 - v_1) & \text{if } \text{Touching}_1 \end{cases} \end{aligned}$$

In the cases where $Touching_1 = \text{False}$, $\dot{P}_1 \leq 0$, therefore $(P_1 \leq 0)$ will be true at least until $(v_1 \leq 0)$ becomes true. If $Touching_1 = \text{True}$ and $v_0 < v_1$ (resp. $v_0 > v_1$) action $Collision_1$ (resp. $Separate_1$) occurs and the trajectory stops. If $Touching_1 = \text{True}$ and $v_0 = v_1$, then $\dot{P}_1 = 0$. Overall, $(P_1 \leq 0) \vee (v_1 \leq 0)$ will be true at the last state of the trajectory. ■

Lemma 4 *If $(P_1 \leq 0) \vee (v_1 \leq 0)$ is true then the pair is safe.*

Proof. If $(v_1 \leq 0)$ is true the pair is safe by Proposition 1. If $(P_1 \leq 0)$, at the time when $\Delta x_1 = 0$, $P_1(\Delta x_1, v_0, v_1) = P_1(0, v_0, v_1) \leq 0$, therefore $(v_0 - v_1)^2 \leq v_A^2$. Hence, $v_1 \leq v_0 + v_A$ and the pair is safe. ■

The conditions of Lemma 4 can be relaxed by introducing P_2 . Consider:

$$I = [P_1 \leq 0] \vee [C \wedge (P_2 \leq 0)] \quad (14)$$

Lemma 5 *$I \vee (v_1 \leq 0)$ is a stable property of the pair.*

Proof. If $(P_1 \leq 0) \vee [C \wedge (P_2 \leq 0)] \vee (v_1 \leq 0)$ is true at the pre-state of $Touch_1$ or $Separate_1$ it will also be true at the post-state as both actions leave $\Delta x_1, v_0$ and v_1 unaffected. Assume $(P_1 \leq 0) \vee [C \wedge (P_2 \leq 0)] \vee (v_1 \leq 0)$ is true when $Collision_1$ occurs. If $(P_1 \leq 0) \vee (v_1 \leq 0)$ is true, it will also be true after $Collision_1$ by Lemma 3. Assume $Collision_1$ occurs while $C \wedge (P_2 \leq 0)$ is true. We distinguish the following cases:

Case 1: $(v_0 = 0) \wedge (P_2 \leq 0)$ is true. Then, at $\Delta x_1 = 0$, $v_1^2 - v_A^2 \leq 0$, therefore $v_1 = v_1 - v_0 \leq v_A$.

Case 2: $(C_1 \leq 0) \wedge (a_0^{min} \leq a_1^{min}) \wedge (P_2 \leq 0)$ is true. Then, $0 < \frac{a_0^{min} + a_1^{min}}{2a_0^{min}} \leq 1$ and at $\Delta x_1 = 0$, $\frac{a_0^{min} + a_1^{min}}{2a_0^{min}} v_0 \geq v_1$. Therefore, $v_0 \geq v_1$ and hence $(C_1 \leq 0) \wedge (a_0^{min} \leq a_1^{min}) \wedge (P_2 \leq 0)$ cannot be true when $Collision_1$ occurs.

Case 3: $(C_2 \leq 0) \wedge (a_0^{min} \geq a_1^{min}) \wedge (P_2 \leq 0)$ is true. This implies that $\frac{a_1^{min}}{a_0^{min}} \geq 1$, $\frac{a_1^{min}}{a_0^{min}} v_0 \leq v_1$ and, at $\Delta x_1 = 0$, $v_1^2 - \frac{a_1^{min}}{a_0^{min}} v_0^2 - v_A^2 \leq 0$. These three inequalities imply that $(v_0 - v_1)^2 - v_A^2 \leq 0$.

In all cases where $Collision_1$ is possible, $0 < v_1 - v_0 \leq v_A$. Therefore $(v_0 - v_1)^2 \leq v_A^2$ and hence $(v_0' - v_1')^2 \leq v_A^2$ (by equation (3) and Assumption 2). Therefore, if $Collision_1$ occurs while $C \wedge (P_2 \leq 0)$ is true, $(P_1 \leq 0)$ will be true after the collision. Overall, if $(P_1 \leq 0) \vee [C \wedge (P_2 \leq 0)] \vee (v_1 \leq 0)$ is true when $Collision_1$ occurs it will also be true afterwards.

Assume at some state, s , $(P_1 \leq 0) \vee [C \wedge (P_2 \leq 0)] \vee (v_1 \leq 0)$ is true and consider the trajectories that start at this state. If $(P_1 \leq 0) \vee (v_1 \leq 0)$ is true at s it will also be true at the last state of the trajectory, by Lemma 3. If $C \wedge (P_2 \leq 0) \wedge (v_1 > 0)$ is true at s , consider the derivatives of the functions C_1, C_2 and P_2 along the trajectory:

$$\frac{d}{dt} C_1 = 2(a_0^{min} + a_1^{min})v_0 acc_0 - 2a_0^{min} acc_0 v_1 - 2a_0^{min} v_0 acc_1 - 2(a_0^{min})^2(v_0 - v_1)$$

$$\begin{aligned}
&= \begin{cases} 0 & \text{if } (v_0 > 0) \wedge \neg \text{Touching}_1 \\ 2(a_0^{\min})^2 v_1 & \text{if } (v_0 = 0) \wedge \neg \text{Touching}_1 \\ 2(a_1^{\min} v_0 - a_0^{\min} v_1) acc_0 - 2(a_0^{\min})^2 (v_0 - v_1) & \text{if } \text{Touching}_1 \end{cases} \\
\frac{d}{dt} C_2 &= \frac{a_1^{\min}}{a_0^{\min}} acc_0 - acc_1 \\
&= \begin{cases} 0 & \text{if } (v_0 > 0) \wedge \neg \text{Touching}_1 \\ -a_1^{\min} & \text{if } (v_0 = 0) \wedge \neg \text{Touching}_1 \\ \left(\frac{a_1^{\min}}{a_0^{\min}} - 1 \right) acc_0 & \text{if } \text{Touching}_1 \end{cases} \\
\frac{d}{dt} P_2 &= 2v_1 acc_1 - 2 \frac{a_1^{\min}}{a_0^{\min}} v_0 acc_0 + 2a_1^{\min} (v_0 - v_1) \\
&= \begin{cases} 0 & \text{if } \neg \text{Touching}_1 \\ 2 \frac{a_0^{\min} v_1 - a_1^{\min} v_0}{a_0^{\min}} acc_0 + 2a_1^{\min} (v_0 - v_1) & \text{if } \text{Touching}_1 \end{cases}
\end{aligned}$$

Consider first the variation of P_2 . If $\text{Touching}_1 = \text{False}$ and as long as $v_1 > 0$, $\dot{P}_2 = 0$. Therefore, if $(P_2 \leq 0)$ is true at s , $(P_2 \leq 0) \vee (v_1 \leq 0)$ will be true at the last state of the trajectory. If $\text{Touching}_1 = \text{True}$ and $v_1 \neq v_0$ the trajectory stops (as the precondition of either Collision_1 or Separate_1 is satisfied). If $\text{Touching}_1 = \text{True}$ and $v_1 = v_0$ then $\dot{P}_2 = 2(a_0^{\min} - a_1^{\min})v_0 acc_0 / a_0^{\min}$. If $a_0^{\min} > a_1^{\min}$ the trajectory stops and action Separate_1 occurs. Otherwise, $\dot{P}_2 \leq 0$, therefore $(P_2 \leq 0)$ will be true at the last state of the trajectory.

Now consider the variation of C . Recall that $C \wedge (v_1 > 0)$ is assumed to be true at s . Distinguish two cases:

Case A: $(C_1 \leq 0) \wedge (a_0^{\min} \leq a_1^{\min})$ is true at s . If $\text{Touching}_1 = \text{False}$ and as long as $v_1 > 0$ and $v_0 > 0$, $\dot{C}_1 = 0$. If $\text{Touching}_1 = \text{True}$ and $v_1 \neq v_0$ the trajectory stops (as the precondition of either Collision_1 or Separate_1 is satisfied). If $\text{Touching}_1 = \text{True}$ and $v_1 = v_0$ then $\dot{C}_1 = 2(a_1^{\min} - a_0^{\min})v_0 acc_0 \leq 0$ as $a_0^{\min} \leq a_1^{\min}$. Overall, $[(C_1 \leq 0) \wedge (a_0^{\min} \leq a_1^{\min})] \vee (v_0 = 0) \vee (v_1 \leq 0)$ will be true at the final state of the trajectory.

Case B: $(C_2 \leq 0) \wedge (a_0^{\min} \geq a_1^{\min})$ is true at s . If $\text{Touching}_1 = \text{False}$ and as long as $v_1 > 0$ and $v_0 > 0$, $\dot{C}_2 = 0$. If $\text{Touching}_1 = \text{True}$ and $v_1 \neq v_0$ the trajectory stops (as the precondition of either Collision_1 or Separate_1 is satisfied). If $\text{Touching}_1 = \text{True}$ and $v_1 = v_0$ then $\dot{C}_2 = (a_1^{\min} - a_0^{\min})acc_0 / a_0^{\min} \leq 0$, as $a_0^{\min} \geq a_1^{\min}$. Therefore, $[(C_2 \leq 0) \wedge (a_0^{\min} \geq a_1^{\min})] \vee (v_0 = 0) \vee (v_1 \leq 0)$ will be true at the final state of the trajectory.

Overall, if $(P_1 \leq 0) \vee [C \wedge (P_2 \leq 0)] \vee (v_1 \leq 0)$ is true at the first state of a trajectory, it will also be true at the last state. ■

Theorem 2 (Sufficient Condition for Pair Safety) *If I is initially true the pair is safe.*

Proof. I initially true and Lemma 5 imply $[P_1 \leq 0] \vee [C \wedge (P_2 \leq 0)] \vee (v_1 \leq 0)$ is an invariant property of the pair. If $(P_1 \leq 0) \vee (v_1 \leq 0)$ is true safety is guaranteed by Lemma 4. If $C \wedge (P_2 \leq 0)$ is true, the proof of Lemma 5 indicates that at $\Delta x_1 = 0$, $v_1 - v_0 \leq v_A$, which again implies safety. ■

Conditions under which the string is unsafe can be obtained in a similar way. Consider a derived boolean variable *Collided* which is initially false and becomes true when the actions Collision_1 occurs. Let:

$$C' = (C_1 \leq 0) \quad (15)$$

$$I' = [-C' \wedge (P_1 > 0)] \vee [(C' \vee (v_0 = 0)) \wedge (P_2 > 0)] \quad (16)$$

Theorem 3 (Necessary Condition for Pair Safety) *If $I' \wedge (v_1 > 0) \wedge \neg \text{Collided}$ is true initially then the pair is unsafe.*

The proof involves an argument similar to the one used for Theorem 2. The proof of Theorem 2 indicates that if the first collision is safe, all subsequent collisions will also be safe. The condition of Theorem 3 is therefore such that the *first* collision between the two vehicles is unsafe. More unsafe collisions may follow.

3.3 Safety Conditions for Strings of Length $N > 2$

Next, we derive a very simple sufficient condition for a string of arbitrary length to be safe. Even though the condition is conservative, interesting conclusions about the safety of platoons of vehicles can be derived from it (see Section 4). A string is *near uniform mass* if $\alpha_i(v) \equiv \alpha$ and $\alpha M_{k-1} \leq M_k \leq M_{k-1}/\alpha$. The near uniform mass condition allows us to put some bounds on the change of speed that a collision can induce. For example, it can be shown that:

Proposition 2 $\bigwedge_{i=0}^{N-1} (v_i \geq 0)$ is an invariant property of a near uniform mass string.

Recall that in general vehicles may end up going backwards due to a collision.

We construct invariant properties that allow us to characterize the safety of such a string. Let $\hat{a}_{min} = \min_{0 \leq k < N} a_k^{min}$ and $\hat{a}_{max} = \max_{0 \leq k < N} a_k^{min}$ and for $0 \leq i < j \leq N-1$ define $\Delta x_{ij} = \sum_{k=i+1}^j \Delta x_k$. For any pair of vehicles $i < j$, consider the function:

$$P(\Delta x_{ij}, v_i, v_j) = v_j - \frac{\hat{a}_{max}}{\hat{a}_{min}} v_i - v_A \quad (17)$$

Theorem 4 (Sufficient Condition for String Safety) *A near uniform mass string of N vehicles is safe if initially $P(\Delta x_{ij}, v_i, v_j) \leq 0$ for all i, j with $0 \leq i < j \leq N-1$.*

The proof is again by induction. Note that the conditions of Theorem 4 involve *all pairs in the string* and not just adjacent vehicles.

Finally, we establish conditions such that any string formed by a collection of vehicles satisfying:

$$a_i^{min} \in [\underline{a}, \bar{a}], \quad M_i \in [\underline{M}, \bar{M}], \quad \alpha_i(v) \equiv 1 \quad (18)$$

is guaranteed to be safe. Assume that all vehicles in the string are initially moving with velocity v .

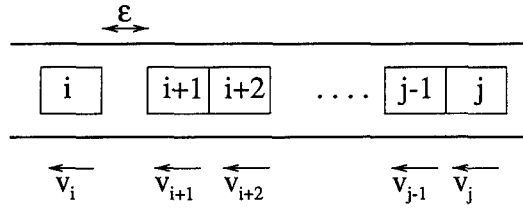


Fig. 3. Final configuration for theorem proof

Theorem 5 (Necessary Condition for String Safety) *All strings of N vehicles satisfying (18) are safe under the default deceleration strategy only if initially $(P_1(\Delta x_{ij}, v, v) \leq 0) \vee (P_2(\Delta x_{ij}, v, v) \leq 0)$ is true for all i, j with $0 \leq i < j \leq N - 1$ and for all $a_i^{min}, a_j^{min} \in [\underline{a}, \bar{a}]$.*

Theorem 5 effectively states that a string may be unsafe if *any two vehicles in it are unsafe*. The proof is constructive: we show that, if two vehicles i and j violate the conditions of the theorem, one can chose the deceleration capabilities, a_k^{min} , and the masses, M_k , of vehicles $k = i + 1, \dots, j - 1$ so that the string exhibits unsafe collisions. The idea of the construction is to bring the vehicles from their initial arrangement to the final arrangement of Figure 3, without any collisions taking place. The construction will be such that after resolving the multiple collision between vehicles $i + 1, \dots, j$ the velocity of vehicle $i + 1$ will be the same as the velocity of vehicle j before the collision. For ϵ small enough, the next collision will be between vehicles $i + 1$ and i and the relative velocity will be ϵ close to the relative velocity with which vehicles j and i would have collided if vehicles $i + 1, \dots, j - 1$ were not there.

4 Implications for Platooning

We establish bounds on the system parameters (in particular the difference in deceleration capability between the vehicles) for a string to be safe. We start with the sufficient condition of Section 3.3. Consider a near uniform mass string and let $\bar{a} - \underline{a} = \epsilon$. Then, all strings whose vehicles satisfy (18) are guaranteed to be safe under the default deceleration strategy if $\left(1 - \frac{\bar{a}}{\underline{a}}\right) v - v_A \leq 0$ or equivalently:

$$\epsilon \leq -\frac{\underline{a}v_A}{v} \quad (19)$$

Substituting "typical" values of $\underline{a} = -9ms^{-2}$ and $v_A = 3ms^{-1}$ leads to $\epsilon \leq 1.08$ for $v = 25ms^{-1}$ and $\epsilon \leq 0.9$ for $v = 30ms^{-1}$.

For the necessary conditions of Section 3.3, note that:

$$\begin{aligned} \frac{\partial P_1}{\partial a_i^{min}} &= -2\Delta x_{ij} \leq 0 & \frac{\partial P_1}{\partial a_j^{min}} &= 2\Delta x_{ij} \geq 0 \\ \frac{\partial P_2}{\partial a_i^{min}} &= \frac{a_j^{min}}{(a_i^{min})^2} v_i^2 \leq 0 & \frac{\partial P_2}{\partial a_j^{min}} &= -\frac{v_i^2}{a_i^{min}} + 2\Delta x_{ij} \geq 0 \end{aligned}$$

N	$\epsilon \text{ (ms}^{-2}\text{)}$		
	$v = 25\text{ms}^{-1}, F = 1m$	$v = 30\text{ms}^{-1}, F = 1m$	$v = 25\text{ms}^{-1}, F = 2m$
2	4.5	4.5	2.25
3	2.25	2.25	1.125
4	1.5	1.5	1.125
5	1.125	1.125	1.125
≥ 6	1.125	0.9	1.125

Table 1. Maximum allowable difference in deceleration capability

Therefore, the condition $(P_1(\Delta x_{ij}, v, v) \leq 0) \vee (P_2(\Delta x_{ij}, v, v) \leq 0)$ for all a_i^{min} and $a_j^{min} \in [\underline{a}, \bar{a}]$ is equivalent to $(P_1(\Delta x_{ij}, v, v) \leq 0) \vee (P_2(\Delta x_{ij}, v, v) \leq 0)$ for $a_i^{min} = \underline{a}$ and $a_j^{min} = \bar{a}$. To further simplify the calculation assume that initially the string is uniformly spaced, i.e. $\Delta x_i = F$ for all i . Then the necessary condition for string safety requires that for all $i \leq j$:

$$\epsilon \leq \max \left\{ \frac{v_A^2}{2(j-i)F}, \frac{2(j-i)\underline{a}^2 F - \underline{a}v_A^2}{v^2 - 2(j-i)\underline{a}F} \right\}$$

Table 1 shows the necessary condition for ϵ . The numbers indicate that the sufficient condition is conservative for small strings but approaches the necessary condition as the string size increases (the number for $N = 2$ in Table 1 is both necessary and sufficient).

If the string represents a platoon and based on the characteristics of vehicles on current highways, the bound on ϵ is reasonable for $N = 2$ but rather restrictive for higher platoon sizes (even under perfect road conditions). Note also that the calculation saturates after the first few vehicles; a similar observation was made in [6] about the increase in deceleration effort required along a platoon for "string stability". Overall, The above calculations indicate that the safety of the platooning system under emergency braking can only be guaranteed under rather limited conditions, in particular for small platoons consisting of vehicles of similar deceleration capabilities. This observation is in agreement with the numerical study of [9]. One can improve the situation by modifying the system parameters, by arranging the vehicles in a platoon in a particular order (e.g. in the order of increasing deceleration capability) and by designing better deceleration controllers. All these alternatives are the topic of current research.

5 Concluding Remarks

The string system introduced here is an interesting example for trying out different hybrid systems techniques. The system is simple enough to approach analytically, yet it can produce executions with very complex continuous-discrete interaction, even for string sizes as small as $N = 3$. Here we used induction

arguments to answer safety questions; induction proofs are ideally suited to the structure imposed by the HIOA modeling formalism used to encode the system.

We are currently working on extending the results discussed here to account for phenomena like sensing and actuation uncertainties and delays. These extensions are likely to involve the use of simulation relations and abstraction mappings (similar analysis was carried out in [5] for a simpler system). We are also trying to investigate the effect of different deceleration strategies. Allowing different deceleration strategies makes the problem much more challenging; for example more sophisticated analysis techniques may be needed to ensure that the proposed controllers do not resort to "Zeno" executions to ensure the safety of the system³. The ultimate goal is of course to construct an optimal deceleration strategy for a each string; powerful optimal control tools are likely to be needed for this purpose. Hopefully solution to these problems will suggest ways in which control theory and computer science techniques can be used in tandem to address complicated questions in hybrid systems.

References

1. N. Lynch, R. Segala, F. Vaandrager, and H. Weinberg, "Hybrid I/O automata," in *Hybrid Systems III*, no. 1066 in LNCS, pp. 496–510, Springer Verlag, 1996.
2. A. Hitchcock, "Casualties in accidents occuring during split and merge maneuvers," tech. rep., PATH Technical Memo 93-9, Institute of Transportation Studies, University of California, Berkeley, 1993.
3. J. Lygeros, D. N. Godbole, and S. Sastry, "A game theoretic approach to hybrid system design," in *Hybrid Systems III*, no. 1066 in LNCS, pp. 1–12, Springer Verlag, 1996.
4. P. Li, L. Alvarez, R. Horowitz, P.-Y. Chen, and J. Carbaugh, "Safe velocity tracking controller for AHS platoon leader," in *IEEE Conference on Decision and Control*, pp. 2283–2288, 1996.
5. E. Dolginova and N. Lynch, "Safety verification for automated platoon maneuvers: a case study," in *Proceedings of HART97* (O. Maler, ed.), no. 1201 in LNCS, pp. 154–170, Berlin: Springer Verlag, 1997.
6. D. Swaroop, *String Stability of Interconnected systems: an application to platooning in automated highway systems*. PhD thesis, Department of Mechanical Engineering, University of California, Berkeley, 1994.
7. J. Lygeros, D. N. Godbole, and S. Sastry, "A verified hybrid controller for automated vehicles," Tech. Rep. UCB-ITS-PRR-97-9, Institute of Transportation Studies, University of California, Berkeley, 1997. (to appear in the Special Issue on Hybrid Systems of the IEEE Transactions on Automatic Control).
8. S. Shladover, "Operation of automated guideway transit vehicles in dynamically re-configured trains and platoons," Tech. Rep. UMTA-MA-0085-79-3, U.S. Department of Transportation, 1979.
9. D. N. Godbole and J. Lygeros, "Tools for safety and throughput analysis of automated highway systems," in *American Control Conference*, pp. 2031–2035, 1997.

³ This is not an issue for the default deceleration strategy considered here, as it is easy to show that all vehicles come to a stop in finite time and after a finite number of collisions.

An Approach to the Verification of the Center-TRACON Automation System*

John Lygeros, George J. Pappas and Shankar Sastry

Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley CA 94720
lygeros,gpappas,sastry@eecs.berkeley.edu

Abstract. The Center-TRACON Automation System (CTAS) is a collection of planning and control software functions that generate landing schedules and advisories to assist air traffic controllers in handling traffic in the en-route and terminal areas. In this paper, we propose a formal safety analysis methodology to determine the correctness of CTAS with respect to safety. Four large classes of safety notions are identified for the CTAS problem: nominal, robust, structural and degraded. For nominal safety questions we seek conditions under which the system is guaranteed to be nominally safe.

1 Introduction

The increasing demand for air travel has spurred the development of tools to increase airspace utilization, smooth air traffic flow and reduce fuel consumption, time delays and controller workload. In an effort to meet these objectives, NASA has developed the Center-TRACON Automation System (CTAS) [1]. CTAS is a collection of planning and control functions which generate advisories to assist, but not replace, the controllers in handling traffic in the Center and TRACON areas.

The structure and functionality of CTAS are briefly discussed in Section 2. CTAS is a large scale, safety critical software system, that should ideally be validated before it is deployed. The validation process is complicated by the fact that the overall system is hybrid, as the (primarily) discrete dynamics of the algorithm are coupled with the continuous dynamics of the aircraft and the human operators. We present a formal approach to the safety analysis of the CTAS system. We view our work as a first step towards the development of a general methodology for the analysis of large scale, hybrid software systems². Our methodology proceeds in the following steps:

* Research supported by the FAA and NASA under Research Contract DTFA03-97-D-0004 and Grant 96-C-001 and by the Army Research Office under Grant DAAH 04-95-1-0588.

² For another example of such a system in air traffic control see [2].

- **System Modeling:** The CTAS system is modeled in the *Hybrid Input-Output Automata* formalism (Section 3).
- **Safety Specification:** We identify notions of safety and determine the desired system specification. We consider four classes of safety measures: *nominal*, *robust*, *structural* and *degraded safety* (Section 4).
- **Safety Analysis:** Given the CTAS model in the hybrid input-output automata formalism and the nominal safety specification, deductive techniques will be used to determine conditions under which the system satisfies the specification. To tackle the complexity of the analysis, high level specifications (at the level of CTAS) are partitioned to sub specifications for the lower level components. The components are analyzed individually to determine whether they meet the corresponding specifications. The proof for the overall CTAS system is composed from the component proofs using abstraction relations.

2 CTAS Overview

The Air Traffic Control system consists of three types of control facilities: Air Route Traffic Control Centers (Centers) which control en-route flights, Terminal Radar and Approach Control facilities (TRACON) which control arriving and departing flights within 30 nautical miles of airports and Airport Control Towers which control traffic in the immediate vicinity of the airport and on the ground. In the United States there are 20 Centers and over 400 TRACONs. The Center-TRACON Automation System provides advisories for the air traffic controllers, in an attempt to increase airspace utilization, reduce delays, fuel consumption and controller workload and improve safety. CTAS consists of three main components:

- **Traffic Management Advisor (TMA)**
- **Descent Advisor (DA)**
- **Final Approach Spacing Tool (FAST)**

TMA [3] and DA [4] coexist and operate in Center airspace whereas FAST [5] operates as a stand alone in TRACON airspace. Even though currently CTAS deals only with arrival traffic, future versions will incorporate the User Preferred Routing tool (UPR) [6] for en route traffic to support Free Flight [7] and the Expedited Departure Path tool (EDP) for departure traffic. UPR and EDP will coexist with TMA and DA in the Center Airspace whereas FAST will be in charge of all terminal area traffic. Currently, stand alone versions of TMA and FAST are being field tested at Dallas-Fort Worth whereas DA is being field tested at Denver.

2.1 CTAS Architecture

The architecture of CTAS is shown in Figure 1. CTAS is a human centered control system. It receives information from the aircraft and computes schedules

and advisories, which are then transmitted to the aircraft by the controllers. The feedback nature of the architecture makes CTAS reactive. If aircraft do not follow the advisories or controllers manually change the landing schedule, CTAS will readjust and produce new advisories in the next computation cycle. Thought of as a large input-output system, CTAS receives input from:

- **Controllers:** The Traffic Management Coordinator (TMC), who resides in a Center, sets the capacity and acceptance rates for various runways, airports and the TRACON and can alter the landing sequence or schedule. The Center and TRACON controllers may select particular routes or runways for particular aircraft and impose constraints on the landing sequence or routing. Controller preferences are inputted through graphical user interfaces: the TMA Graphical User Interface (TGUI), which is used by the TMC, and the Plainview Graphical User Interface (PGUI), which is used by all Center and TRACON controllers.
- **Radar Daemon:** The radar daemon periodically receives aircraft state information regarding position, altitude, speed, aircraft type and flight plan.
- **Weather Daemon:** The weather daemon receives weather information from the National Weather Service. Currently the weather reports include wind, temperature and pressure profiles in the form of a three dimensional grid whose edge length is 50 miles. The weather reports are updated every hour and contain forecasts for the next three hours.

Internally, CTAS utilizes detailed databases which include aircraft, aerodynamic and engine models for all aircraft types. These models are used to perform accurate trajectory prediction for each aircraft. Other databases contain information on the local airspace structure in terms of way-points and routes, site adaptation data such as TRACON, airport and runway configurations and site specific constraints. A Communication Manager supports the internal communication of data between the various processes. After all internal calculations are performed, the main outputs of CTAS are:

- **Landing Schedules:** CTAS performs runway allocation for all arriving aircraft and produces a time-line schedule and sequence for each runway. Scheduling is initially performed in the Center Area by the TMA and the output is graphically displayed on the TGUI which is used by the TMC. Once in the TRACON area, FAST recomputes and overrides the previous schedule. The new schedule is then displayed on the PGUI.
- **Advisories:** CTAS also computes heading, altitude and speed advisories. DA provides the advisories in Center airspace whereas FAST provides advisories in the terminal area. The advisories are displayed on the GUIs and are then transmitted by voice from the controllers to the aircraft.

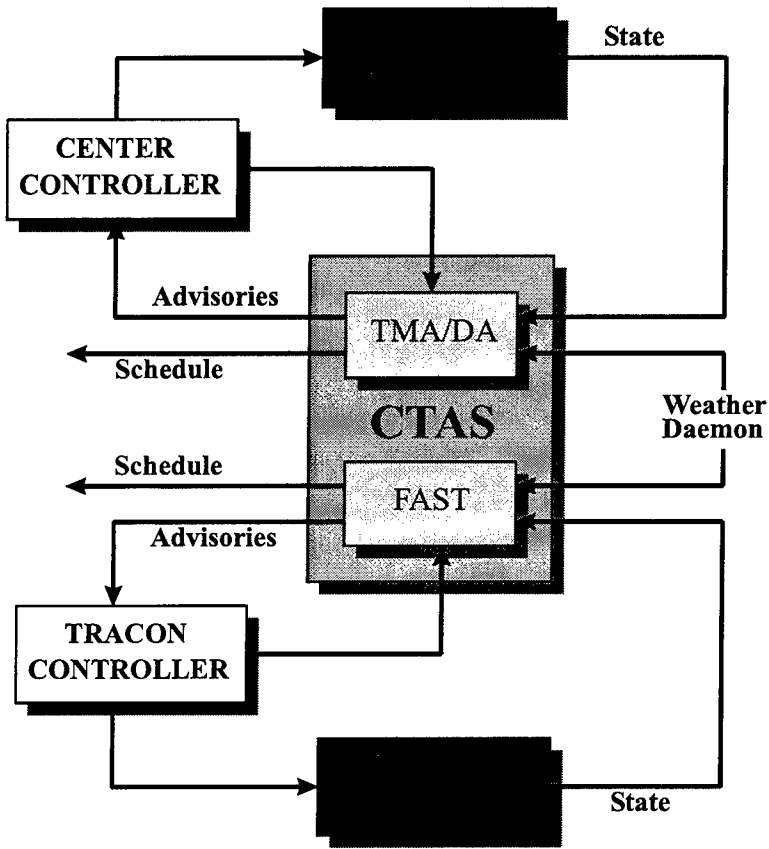


Fig. 1. CTAS Architecture

3 Hybrid Input-Output Automata

In this section we give a brief overview of a model for the CTAS system. In Section 3.1 we outline the modeling formalism (a formal discussion can be found in [8]) and discuss the features that make it ideal for modeling the CTAS system. In Section 3.2 we show how models can be constructed in this framework for one of the CTAS components, the FAST algorithm.

3.1 Overview of the Modeling Framework

Based on the work of [8], we consider a hybrid automaton, A , as a dynamical system that describes the evolution of a finite collection of variables, V_A . Variables are typed; for each $v \in V_A$ let $type(v)$ denote the type of v . For each $Z \subseteq V_A$, a *valuation* of Z is a function that to each $v \in Z$ assigns a value in $type(v)$. Let \mathbf{Z} denote the set of valuations of Z ; we refer to $s \in \mathbf{V}_A$ as a system

state. In this paper we assume that the evolution of the variables is over the set $T^{\geq 0} = \{t \in \mathbb{R} | t \geq 0\}$. The evolution of the variables involves both continuous and discrete dynamics. Continuous dynamics are encoded in terms of *trajectories* over V_A , that is functions that map intervals of the time axis to V_A . Discrete dynamics are encoded by *actions*. Upon the occurrence of an action the system state instantaneously “jumps” to a new value. We use Σ_A to denote the set of actions that affect the evolution of A .

Formally, a *hybrid automaton*, $A = (U_A, X_A, Y_A, \Sigma_A^{in}, \Sigma_A^{int}, \Sigma_A^{out}, \Theta_A, \mathcal{D}_A, \mathcal{W}_A)$, is a collection of:

- Three disjoint sets U_A , X_A , and Y_A of variables, called *input*, *internal*, and *output variables*, respectively. We set $V_A = U_A \cup X_A \cup Y_A$.
- Three disjoint sets Σ_A^{in} , Σ_A^{int} , and Σ_A^{out} of actions, called *input*, *internal*, and *output actions*, respectively. We set $\Sigma_A = \Sigma_A^{in} \cup \Sigma_A^{int} \cup \Sigma_A^{out}$.
- A non-empty set $\Theta_A \subseteq V_A$ of *initial states*.
- A set $\mathcal{D}_A \subseteq V_A \times \Sigma_A \times V_A$ of *discrete transitions*.
- A set \mathcal{W}_A of *trajectories* over V_A .

Some technical axioms are imposed on the above sets to guarantee that the definitions are consistent.

An *execution*, α , of the hybrid automaton A is a finite or infinite alternating sequence $\alpha = w_0 a_1 w_1 a_2 w_2 \dots$, where for all i , $a_i \in \Sigma_A$, $w_i \in \mathcal{W}_A$ defined over a left closed time interval, $fstate(w_0) \in \Theta_A$, if α is a finite sequence then it ends with a trajectory and if w_i is not the last trajectory its domain is right-closed and $(lstate(w_i), a_{i+1}, fstate(w_{i+1})) \in \mathcal{D}_A$. Here $fstate(w)$ and $lstate(w)$ denote the initial and final states of a trajectory w . An execution is called *finite* if it is a finite sequence and the domain of its final trajectory is a right-closed interval. A state $s \in V_A$ is called *reachable* if it is the last state of a finite execution.

To capture the evolution of an HIOA from the point of view of the “outside world” the notion of a *trace* is introduced. Roughly speaking a trace is an execution projected to the external (input and output) variables and actions. Two automata A and B are called *comparable* if they have the same external interface. If A and B are comparable, then we say that A *implements* B if the hybrid traces of A are a subset of those of B . Typically one thinks of B as a *specification* and A as an *implementation* of the specification. A specification is usually a more abstract description that imposes weaker restrictions on the system behavior. Proving that one hybrid automaton implements another can be a complicated task. Usually it is broken up in a series of steps, where the abstract specification is progressively refined. At each step implementation is proved using simulation relations. A *simulation* from A to B is a relation $R \subseteq V_A \times V_B$ such that (roughly speaking) two states, r of A and s of B , are related through R if from state s B can reproduce any move that A makes from r (discrete or continuous) by a hybrid execution which is indistinguishable from the move of A from the point of view of the outside world. The final states of the two moves should again be related by R .

Hybrid automata “communicate” through shared variables and shared actions. Consider two automata A and B with $X_A \cap V_B = X_B \cap V_A = Y_B \cap Y_A = \emptyset$

and $\Sigma_B^{int} \cap \Sigma_A = \Sigma_A^{int} \cap \Sigma_B = \Sigma_A^{out} \cap \Sigma_B^{out} = \emptyset$. Under some mild technical assumptions, the *composition*, $A \times B$, of A and B can be defined as a new hybrid automaton with $U_{A \times B} = (U_A \cup U_B) \setminus (Y_A \cup Y_B)$, $X_{A \times B} = X_A \cup X_B$, $Y_{A \times B} = Y_A \cup Y_B$ (similarly for the actions). $\Theta_{A \times B}$, $\mathcal{D}_{A \times B}$ and $\mathcal{W}_{A \times B}$ are such that the executions of $A \times B$ are also executions of each automaton when restricted to its variables and actions. It can be shown that composition respects implementation.

A *derived variable* of A is a function on \mathbf{V}_A . Derived variables will be used to simplify the system description, but also to facilitate the analysis. A *property* of A is a boolean derived variable. A property is *stable* if whenever it is true at some state it is also true at all states reachable from that state. A property is *invariant* if it is true at all reachable states. Typically properties will be shown to be stable or invariant by an induction argument on the length of an execution.

In some places differential equations will be used to simplify the description of the set \mathcal{W}_A . In such cases \mathcal{W}_A is assumed to be populated by all trajectories generated by the differential equation in the usual way. To simplify the description of \mathcal{D}_A , we will assign a *precondition* and an *effect* to each action. The precondition is a predicate on \mathbf{V}_A while the effect is a predicate on $\mathbf{V}_A \times \mathbf{V}_A$. The action can take place only from states that satisfy the precondition; moreover, the states before and after the transition should be such that the effect is satisfied.

The following properties of the HIOA formalism are especially useful for CTAS modeling:

1. **Descriptive Power:** The modeling formalism provides a uniform framework in which one can describe the evolution of general classes of variables, ranging from real and integer numbers (with their associated mathematical structure) to abstract high level data types typically found in a computer program.
2. **Hybrid Dynamics:** The formalism allows us to capture continuous and discrete dynamics and the interaction between the two.
3. **Compositionality:** The modeling formalism allows us to build up the description of the complicated CTAS system by combining simpler entities.
4. **Abstraction:** The notion of specification and implementation allows us to describe the system at various levels of abstraction. This provides a way of showing that the CTAS code satisfies a specification through a sequence of progressive refinement. Abstraction also allows us to structure proofs hierarchically, with simulation relations connecting the levels of the proof hierarchy.

3.2 Formal CTAS Model

The CTAS system will be modeled as an interconnection of a number of components (Figure 2). In this section we will show how the input-output interaction between these components can be captured by appropriate hybrid automata. The models given here will be “high level”; in a number of places we will use

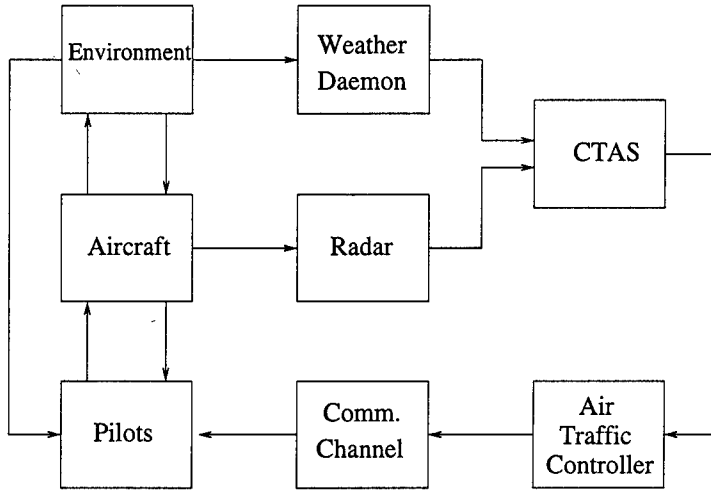


Fig. 2. CTAS Feedback Loop

derived variables to represent parts of the algorithm, the air traffic controller behavior, etc. for which we can not provide explicit expressions at this stage. The model can be refined until it contains sufficient details to carry out meaningful safety analysis. The refinement primarily involves providing explicit expressions for these derived variables. Here we will briefly discuss the operation of all the components shown in Figure 2. For the time being we restrict our attention to FAST; the remaining CTAS components can be similarly modeled. Examples of HIOA pseudo-code for FAST can be found in the appendix; for more examples see [9].

Aircraft Model: The system we consider consists of N aircraft, labeled $1, \dots, N$. Each aircraft, i , is modeled by a hybrid automaton, $A_i = (U_{A_i}, X_{A_i}, Y_{A_i}, \Sigma_{A_i}^{in}, \Sigma_{A_i}^{int}, \Sigma_{A_i}^{out}, \Theta_{A_i}, \mathcal{D}_{A_i}, \mathcal{W}_{A_i})$. At this stage we assume that the aircraft evolution is not affected by any actions, input, output or internal ($\Sigma_{A_i} = \Sigma_{A_i}^{in} \cup \Sigma_{A_i}^{int} \cup \Sigma_{A_i}^{out} = \emptyset$ and hence $\mathcal{D}_{A_i} = \emptyset$). At a later stage appropriate actions can be added to model discrete changes in the physical system, such as malfunctions.

Each aircraft is identified by its type, for example, TurboJet, 747, etc. This information is stored in an internal variable $Type_i$. The physical movement of the aircraft is summarized by the trajectories of its position and velocity. Let $p_i = (x_i, y_i, z_i) \in \mathbb{R}^3$, $v_i = (v_i^x, v_i^y, v_i^z) \in \mathbb{R}^3$ be the position and velocity of the aircraft with respect to some fixed reference frame on the ground. The motion of the aircraft is influenced by the commands of the pilot and the environmental conditions. Let a_i represent the pilot commands (for the engine, control surfaces, flaps, etc.) and w_i the environmental conditions at the current position of aircraft i (wind and temperature for example). We assume that all trajectories in \mathcal{W}_{A_i}

satisfy the differential equation:

$$\begin{bmatrix} \dot{p}_i(t) \\ \dot{v}_i(t) \end{bmatrix} = \begin{bmatrix} v_i(t) \\ f(\text{Type}_i, v_i(t), a_i(t), w_i(t)) \end{bmatrix} \quad (1)$$

We set $Y_{A_i} = \{\text{Type}_i, p_i, v_i\}$, $U_{A_i} = \{a_i, w_i\}$ and $X_{A_i} = \emptyset$. The function f returns the acceleration of the aircraft, which will in general depend on the aircraft type, the aircraft velocity, the commands of the pilot and the weather conditions.

The aircraft automaton is only partly specified at this stage. We still need to provide an expression for the derived variable f . This expression is likely to be very complicated. For the preliminary safety analysis we start with simple formulas (such as $f = a_i$); more accurate expressions can be obtained from the aircraft model database used by CTAS.

Environment Model: We assume that the motion of each aircraft is influenced by the wind and temperature at its current position. To encode this information we introduce a hybrid automaton $E = (U_E, X_E, Y_E, \Sigma_E^{in}, \Sigma_E^{int}, \Sigma_E^{out}, \Theta_E, \mathcal{D}_E, \mathcal{W}_E)$. The environment automaton has no internal, input or output actions ($\Sigma_E = \emptyset$) and no internal variables ($X_E = \emptyset$). Its inputs are the positions of all aircraft, ($U_E = \{p_i\}_{i=1}^N$) and its outputs (denoted by $w_i \in \mathbb{R}^4$) are the wind magnitude and direction and the temperature at each one of these positions ($Y_E = \{w_i\}_{i=1}^N$). The environmental conditions are encoded by a function $W : \mathbb{R}^+ \times \mathbb{R}^3 \rightarrow \mathbb{R}^4$, that returns the wind and temperature at the current time and the given location.

The environment model is only partly specified at this stage. To complete the description we need to provide an expression for the derived variable W . We propose to start with very simple expressions (e.g. W constant as a function of time) and refine them at a later stage, as the description of the automata that make use of the weather information n (the aircraft and the weather daemon soon to be specified) becomes more detailed.

Radar Automaton: CTAS obtains information about the state of each aircraft through radar. We model the radar by a hybrid automaton $R = (U_R, X_R, Y_R, \Sigma_R^{in}, \Sigma_R^{int}, \Sigma_R^{out}, \Theta_R, \mathcal{D}_R, \mathcal{W}_R)$. The input variables of R are the positions and velocities of all aircraft ($U_R = \{p_i, v_i\}_{i=1}^N$) while the output variables of R are estimates of these quantities, denoted by \hat{p}_i and \hat{v}_i ($Y_R = \{\hat{p}_i, \hat{v}_i\}_{i=1}^N$). At this stage the radar automaton is assumed to have no input or internal actions ($\Sigma_R^{in} = \Sigma_R^{int} = \emptyset$).

The information that the radar provides about the aircraft is quantized spatially and sampled temporally. We assume that the output variables of the radar automaton fall within an interval centered at the "correct" values dictated by the actual state of the system. Let $n_P \in \mathbb{R}^3$ and $n_V \in \mathbb{R}^3$ denote the width of the intervals for \hat{p}_i and \hat{v}_i respectively. At this stage we assume n_P and n_V are constant; these quantities may become internal variables later on, to model variations of the accuracy of sensing with position and environmental conditions, for example. The output variables of the radar are updated every \hat{T}_r seconds, upon the occurrence of an output action Sample_r . We set $\Sigma_R^{out} = \{\text{Sample}_r\}$. An internal variable $T_r \in \mathbb{R}$ keeps track of the time that has elapsed since the

last sample. \hat{T}_r is typically of the order of a few seconds. Once values for n_P , n_V and \hat{T}_r are available the radar automaton will be completely specified.

Weather Daemon: CTAS also obtains information about the environmental conditions in the vicinity of each aircraft. This information is provided by a hybrid automaton $D = (U_D, X_D, Y_D, \Sigma_D^{in}, \Sigma_D^{int}, \Sigma_D^{out}, \Theta_D, \mathcal{D}_D, \mathcal{W}_D)$. D is very similar to the radar automaton R . D has no input or internal actions ($\Sigma_D^{in} = \Sigma_D^{int} = \emptyset$). Its input variables are the weather conditions at the location of each aircraft ($U_R = \{w_i\}_{i=1}^N$) and its output variables are estimates of these quantities denoted by \hat{w}_i ($Y_R = \{\hat{w}_i\}_{i=1}^N$). \hat{w}_i are quantized to within $n_w \in \mathbb{R}^4$ of the real environmental conditions and sampled every \hat{T}_w time units, upon the occurrence of an output action $Sample_w$.

The FAST Automaton: The FAST algorithm itself will be encoded by a hybrid automaton $FAST = (U_{FAST}, X_{FAST}, Y_{FAST}, \Sigma_{FAST}^{in}, \Sigma_{FAST}^{int}, \Sigma_{FAST}^{out}, \Theta_{FAST}, \mathcal{D}_{FAST}, \mathcal{W}_{FAST})$. The input variables of $FAST$ are the output variables of the radar and weather daemon automata and type information from the aircraft automata. Overall, $U_{FAST} = Y_R \cup Y_D \cup \{Type_i\}_{i=1}^N$. $FAST$ provides advisories to the air traffic controllers for all aircraft, $Y_{FAST} = \{adv_i\}_{i=1}^N$. An internal variable, $FAST_AC \subset \{1, \dots, N\}$, is used to store the labels of all aircraft currently in the TRACON. It is assumed that for aircraft not currently in the TRACON ($i \notin FAST_AC$) a default advisory $adv_i = \perp$ (undefined) is issued. For aircraft in the TRACON, the nature of the advisory depends on the version of FAST. This information is stored in an internal variable $Version \in \{\text{Active}, \text{Passive}\}$. If $Version = \text{Passive}$, the advisory for each aircraft consists of a runway assignment and a landing sequence. If $Version = \text{Active}$, FAST also provides heading, altitude or speed commands. At this stage we model these commands as a position in the $x - y$ plane, P , and a number, V , that encodes a desired heading altitude or speed; the interpretation is that the pilot is asked to guide the aircraft to P and achieve V by the time it gets there. The advisory calculations involve the degrees of freedom available to each aircraft and are restricted by sequencing constraints imposed by the air traffic controllers. This information is stored in internal variables dof_i and $Constraints$. Overall, $X_{FAST} = \{FAST_AC, Constraints, Version\} \cup \{dof_i\}_{i=1}^N$.

The evolution of the FAST automaton is disrupted by input actions. After each action the advisories for all aircraft currently in the TRACON are recalculated. The calculation is encoded by a derived variable $Calculate_Advisory$. Input actions $Sample_r$ and $Sample_w$ are the output actions of the radar and weather daemon automata respectively. Their role is to recalculate the advisories whenever new data becomes available. Input action $Center_Handoff(i)$ occurs when aircraft i enters the TRACON. It represents the hand-off of aircraft from the Center to the TRACON air traffic controllers and is assumed to be the output of a hybrid automaton modeling the air traffic controllers. The effect of $Center_Handoff(i)$ is to add aircraft i to the list of aircraft currently in the TRACON, initialize its possible degrees of freedom and recalculate the advisories for all aircraft. The degrees of freedom are initialized according to a derived variable $Default_dof(i)$, whose "output" will typically depend on the

aircraft type, the point of entry into the TRACON, the weather, etc.

The TRACON air traffic controller influences the evolution of FAST through three input actions. Using action *Constrain_Order*(i, j), the controller can force FAST to schedule aircraft i before aircraft j . The effect of this action is to add (i, j) to the list of sequencing constraints maintained by FAST and recalculate the advisories. *Constrain_dof*(i, dof) allows the controller to reduce the degrees of freedom that FAST considers for aircraft i . Upon occurrence of the action FAST removes the specified degree of freedom from the list dof_i and recalculates the advisories. Finally, the action *Tower_Handoff*(i) occurs when aircraft i lands and is handed-off to the tower controllers. The effect of the action is to remove i from the list of aircraft currently in the TRACON, together with all sequencing constraints involving i on the remaining aircraft. The advisories for the remaining aircraft are recalculated.

To complete the description of the FAST automaton we need to provide expressions for the derived variables *Default_dof*(i) and *Calculate_Advisory*. The expressions need to be extracted from the FAST documentation.

Air Traffic Controller Model: The air traffic controllers are modeled by a hybrid automaton, $ATC = (U_{ATC}, X_{ATC}, Y_{ATC}, \Sigma_{ATC}^{in}, \Sigma_{ATC}^{int}, \Sigma_{ATC}^{out}, \Theta_{ATC}, \mathcal{D}_{ATC}, \mathcal{W}_{ATC})$. The inputs to ATC are the advisories from FAST as well as all the information available for each aircraft. Overall $U_{ATC} = \{\text{adv}_i, \text{Type}_i, \hat{p}_i, \hat{v}_i, \hat{w}_i\}_{i=1}^N$. As at this stage we are only concerned with the FAST operation, ATC will primarily model the TRACON air traffic controllers. The only function of the Center air traffic controllers in this setting is to feed aircraft into the TRACON, by executing action *Center_Handoff*(i). We assume that the Center contains a number of aircraft, whose labels are stored in an internal variable *Center_AC*. An aircraft gets removed from this list and is handed off to the TRACON controller upon the occurrence of output action *Center_Handoff*(i). The precondition of the action is a boolean derived variable *Center_Handoff_Condition*(i).

The TRACON controller may choose not to follow a particular advisory or to follow it after some delay. This information is stored in the boolean internal variables *Follow* $_i$ and the real internal variables d_i . We assume that the controller keeps track of the previous advisory issued by CTAS for aircraft i in an internal variable *Old_Advisory* $_i$. *Old_Advisory* $_i$ is used to trigger an internal action *New_Advisory* $_i$. Upon occurrence of the action the controller decides whether the new advisory will be followed and selects a delay. If the controller chooses to follow the advisory, the speed, altitude or heading command is transmitted to the pilot of aircraft i after a delay d_i , upon the occurrence of an output action *Send* $_i$. If the controller chooses not to follow the advisory or if FAST is "passive" the transmitted command is assumed to be determined by a derived variable *Independent_Choice*(i). Our model also allows controllers to issue independent commands in between the FAST advisories, whenever a boolean derived variable *Independent_Choice_Condition*(i) becomes true.

The controller can influence FAST through actions *Constrain_Order*(i, j) and *Constrain_dof*(i, dof). The preconditions for these actions are encoded by boolean

internal variables $Order_Condition(i, j)$ and $DOF_Condition(i, dof)$. We assume that the controller keeps track of the constraints it has previously issued. This will allow us to make the controller model more realistic later on (for example, require that the controller does not issue contradictory constraints). Finally, the TRACON controller decides when the aircraft has landed and hands it off to the tower controllers. This "action" is encoded by $Tower_Handoff$. The precondition for this action is a boolean derived variable $Tower_Handoff_Condition(i)$.

The controller model requires expressions for $Center_Handoff_Condition(i)$, $Order_Condition(i, j)$, $DOF_Condition(i, dof)$, $Independent_Choice_Condition(i)$, $Independent_Choice(i)$ and $Tower_Handoff_Condition(i)$. Obtaining expressions for these variables is likely to be a major challenge, as it involves understanding the complicated decision making process of the human air traffic controllers. To start the safety analysis we will assume that FAST is active, the controller always follows the proposed advisories, never imposes additional constraints and hands off the aircraft to the tower at the runway threshold.

Communication Channel: Communicating commands to the pilots is achieved through communication channel automata, C_i . Each automaton has an input action $Send_i(command)$, whose effect is to store the command together with a time stamp in an internal multi-set. The message is delivered (and removed from the multi set) upon occurrence of the output action $Receive_i(command)$. Delivery is guaranteed by at most d_i^c time units from the time the message was sent.

Pilot Model: Finally, the pilot is modeled by a hybrid automaton P_i . P_i accepts input information about the aircraft and the air traffic controller commands (obtained through the input action $Receive_i(command)$) and produces input a_i for the aircraft automaton. Similar to the air traffic controllers, a pilot is given the freedom to ignore an ATC command. His/her decision is stored in an internal variable $Follow_i^p$. If the pilot chooses to follow a particular command he/she responds after some delay (encoded by input variable d_i^p). In this case, a_i is chosen according to a derived variable $Comply$. Otherwise, a_i is chosen according to a derived variable Not_Comply . Expressions for these derived variables are needed to complete the description of the pilot automaton. These expressions may again be hard to obtain as they involve modeling the response of the human pilots and/or the autopilots.

4 Safety Notions

The performance evaluation of large scale systems like CTAS is a very complex process. Various metrics quantitatively measure system performance and allow comparisons between different designs. The three most prominent performance areas for CTAS are:

- **Safety**, which receives top priority
- **Economic considerations**, such as minimizing fuel and operating costs as well as time delays. Other considerations, such as passenger comfort can also be included in this category.

- **Reduction of controller workload** and, more generally, increasing situational awareness of controllers.

Even though all three aspects of the system performance are important, and the interaction between them is very interesting, here we will concentrate on questions of safety. We classify of safety questions into:

- **Nominal Safety:** considers safety under nominal conditions
- **Robust Safety:** questions the robustness of the nominal safety claims,
- **Structural Safety:** questions of safety under structural changes in CTAS
- **Degraded Safety:** considers safety questions in degraded operation.

The above classes of safety measures will be used to determine not whether CTAS is safe but whether CTAS is safer than the current system. The outcome may also depend on the metric used. For example, CTAS may be safer than the current system under nominal operation but not as safe in degraded operation.

For the time being we restrict our attentions to safety questions when the system operation is nominal (in a sense “perfect”). We assume that operation is nominal if:

- **Nominal CTAS:** We start with fixed and reliable version of the CTAS algorithms.
- **Faultless Operation:** There are no hardware malfunctions, no emergency situations (such as aircraft low on fuel), and the environmental conditions are benign.
- **Accurate Models:** The models used by CTAS can accurately predict aircraft movement. This includes the aircraft dynamical models and the weather models. In addition there is no uncertainty in sensors or parameters. For nominal analysis both controllers and pilots can be modeled by a variable delay that nondeterministically takes values in a bounded interval.

Under nominal conditions we can ask the following very precise safety questions which can be thought of as the CTAS nominal safety specification:

- **Completeness:** Will CTAS issue an advisory in every situation?
- **Consistency:** Will CTAS issue the same advisory in identical situations? Consistency is related to controller workload since system predictability increases situational awareness.
- **Stability:** Are the CTAS outputs stable? This is also related to controller workload since advisory changes reduce situational awareness.
- **Separation Requirements:** Loss of separation could be catastrophic and cannot be tolerated.
- **Implementability:** Are the CTAS advisories implementable? Do CTAS advisories satisfy constraints imposed by aircraft dynamics (e.g. stall conditions)?
- **Delay:** What is the effect of delay (in the radar, weather daemon, controller and pilot responses) in the system?

- **Capacity Limits:** What is the maximum possible TRACON capacity or runway acceptance rate for which CTAS can maintain safety? This is related to cost/benefit analysis.

The above list of high level CTAS specifications is refined to the lower levels of the hierarchical structure shown in Figure 3, to derive nominal safety specifications for the CTAS subsystems. Specifications at the level of TMA, DA and FAST can be further decomposed into specifications for lower subsystems and functions (the Route Analyzer (RA), Trajectory Synthesizer (TS), Profile Selector (PFS), Dynamic Planner (DP), etc.) resulting in a set of nominal safety specifications for each component.

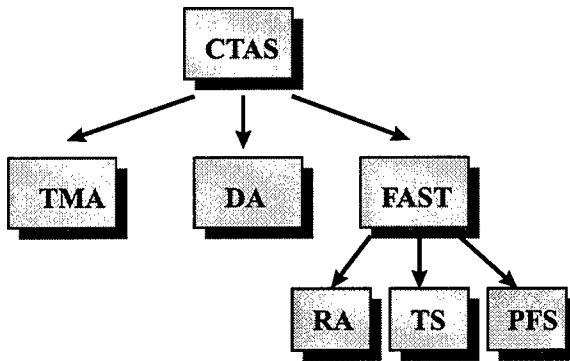


Fig. 3. CTAS Hierarchical Specification Refinement

Whether CTAS satisfies the specifications will depend on the initial configuration and the system parameters. *Our safety analysis methodology for nominal safety will determine the range of configurations and parameter values for which the CTAS advisories are safe.* For example, this will involve determining the rate at which FAST can accept and safely land aircraft that enter through the TRACON gates, for a given runway configuration. As the flow of aircraft to the TRACON gates is determined by the Center TMA, the TMA must in turn guarantee that this flow constraint is not violated. Nominal safety notions try to determine conditions under which the nominal system meets the desired specification. In general the more relaxed the conditions are, the safer the nominal system is. For example, if CTAS can safely handle a flow rate of 60 aircraft an hour in the TRACON under nominal conditions, then it is likely to be more robust than a similar system that can safely handle 50 aircraft an hour.

Given the nominal safety specifications for the various CTAS systems, the next three classes of safety notions try to measure the effect of uncertainty, structural changes and failures to the nominal safety issues.

5 Conclusions

In this paper, a framework for the modeling, specification and safety analysis of the Center-TRACON Automation System (CTAS) is proposed. We believe that this "system theoretic" perspective can prove very fruitful not only for the CTAS problem, but also more generally for the verification of complex, hybrid software systems (see for example [2] for the application of this methodology to the Traffic Alert and Collision Avoidance System (TCAS)). The discussion presented in this paper is only a first step in the verification process of CTAS. Some of the safety questions we formulate are challenging and may require extending the state-of-the-art analysis and verification techniques.

Acknowledgments: The authors would like to thank Darren Cofer and Rosa Weber from Honeywell Technology Center and Nancy Lynch from the Laboratory for Computer Science at MIT for their contributions at various phases of this project.

References

1. H. Erzberger, T.J. Davis, and S. Green. Design of center-TRACON automation system. In *Proceedings of the AGARD Guidance and Control Symposium on Machine Intelligence in Air Traffic Management*, Berlin, Germany, May 1993.
2. John Lygeros and Nancy Lynch. Towards the formal verification of the TCAS conflict resolution algorithms. In *IEEE Control and Decision Conference*, pages 1829-1834, 1997.
3. W. Nedell and H. Erzberger. The traffic management advisor. In *American Control Conference*, San Diego, CA, December 1990.
4. Steven M. Green, Robert A. Vivona, and Beverly Sanford. Descent advisor preliminary field test. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, Baltimore, MD, August 1995.
5. T.J. Davis, K.J. Krzczowski, and C. Bergh. The final approach spacing tool. In *Proceedings of the 13th IFAC Symposium on Automatic Control in Aerospace*, Palo Alto, CA, September 1994.
6. S. Green, T. Goka, and D.H. Williams. Enabling user preferences through data exchange. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, New Orleans, LA, August 1997.
7. Radio Technical Commission for Aeronautics. Final report of RTCA Task Force 3: Free Flight Implementation. Technical report, RTCA, Washington, DC, October 1995.
8. Nancy Lynch, Roberto Segala, Frits Vaandrager, and H.B. Weinberg. Hybrid I/O automata. In *Hybrid Systems III*, number 1066 in LNCS, pages 496-510. Springer Verlag, 1996.
9. George Pappas, John Lygeros, Shankar Sastry, and Nancy Lynch. Modeling, specification and safety analysis of CTAS. Technical Report NEXTOR Research Report RR-97-5, University of California at Berkeley, Berkeley, CA, September 1997.

A FAST Automaton Pseudo-Code

Data Types:

Runways = $\{17L, 17R, \dots\}$
 Types = $\{\text{TurboJet}, 747, DC-10, \dots\}$
 Aircraft = $\{1, \dots, N\} \subset \mathbb{N}$
 Commands = $\{(P, V, K)\}$ with $P \in \mathbb{R}^2$, $V \in \mathbb{R}$, $K \in \{\text{Heading, Speed, Altitude}\}$
 Commands $_{\perp}$ = Commands $\cup \{\perp\}$
 Advisories = $\{(r, s, c)\}$ with $r \in \text{Runways}$, $s \in \{1, \dots, N\}$, $c \in \text{Commands}_{\perp}$
 Advisories $_{\perp}$ = Advisories $\cup \{\perp\}$
 Weather = $\{(\text{Wind, Temperature})\} \subset \mathbb{R}^4$

Variables:

Input:

$\hat{w}_i \in \text{Weather}$ for all $i \in \text{Aircraft}$
 $\hat{p}_i \in \mathbb{R}^3$ for all $i \in \text{Aircraft}$
 $\hat{v}_i \in \mathbb{R}^3$ for all $i \in \text{Aircraft}$
 $\text{Type}_i \in \text{Types}$ for all $i \in \text{Aircraft}$

Internal:

$\text{FAST_AC} \subset \text{Aircraft}$, initially \emptyset
 $\text{dof}_i \subset \text{DOF}$, initially \emptyset
 $\text{Constraints} \subset \text{Aircraft} \times \text{Aircraft}$, initially \emptyset
 $\text{Version} \in \{\text{Active, Pasive}\}$, initially arbitrary

Output:

$\text{adv}_i \in \text{Advisories}_{\perp}$ for all $i \in \text{Aircraft}$, initially \perp

Derived:

$\text{Default_dof}(i) \subset \text{DOF}$, for all $i \in \text{Aircraft}$
 $\text{Calculate_Advisory} \in \text{Bool}$

Actions:

Input:

e , the environment action
 $\text{Center_Handoff}(i)$, $i \in \text{Aircraft}$
 $\text{Constrain_Order}(i, j)$, $i, j \in \text{Aircraft}$
 $\text{Constrain_dof}(i, \text{dof})$, $i \in \text{Aircraft}$, $\text{dof} \in \text{DOF}$
 $\text{Tower_Handoff}(i)$, $i \in \text{Aircraft}$
 Sample_s
 Sample_w

Discrete Transitions:

e :

Effect: arbitrarily reset the input variables

$\text{Center_Handoff}(i)$:

Effect:

$FAST_AC := FAST_AC_i \cup \{i\}$

$dof_i := Default_dof(i)$

for $j \in FAST_AC$, choose adv_j so that *Calculate_Advisory* becomes true

Constrain_Order(i, j):

Effect:

$Constraints := Constraints \cup \{(i, j)\}$

for $j \in FAST_AC$, choose adv_j so that *Calculate_Advisory* becomes true

Constrain_dof(i, dof):

Effect:

$dof_i := dof_i \setminus \{dof\}$

for $j \in FAST_AC$, choose adv_j so that *Calculate_Advisory* becomes true

Tower_Handoff(i):

Effect:

$FAST_AC := FAST_AC \setminus \{i\}$

$dof_i := \emptyset$

$Constraints := Constraints \setminus \bigcup_{j=1}^N (\{(i, j)\} \cup \{(j, i)\})$

for $j \in FAST_AC$, choose adv_j so that *Calculate_Advisory* becomes true

Sample_s and *Sample_w*:

Effect:

for $j \in FAST_AC$, choose adv_j so that *Calculate_Advisory* becomes true

Trajectories:

Input variables follow arbitrary trajectories

Output variables remain constant

Trajectories stop once the precondition of *Tower_Handoff*(i) becomes true

Deductive Verification of Hybrid Systems Using STeP *

Zohar Manna and Henny B. Sipma
manna|sipma@cs.stanford.edu

Computer Science Department, Stanford University
Stanford, CA 94305

Abstract. We investigate the feasibility of computer-aided deductive verification of hybrid systems. Hybrid systems are modeled by phase transition systems, in which activities specify the bounds on the derivatives of the continuous variables. We present a method for invariant generation based on static analysis of the phase transition system. The invariants produced can be used as auxiliary properties in the verification of temporal properties. We show that in some cases the invariants thus produced suffice to prove the main safety property.

1 Introduction

Deductive approaches to the verification of hybrid systems have been studied extensively. However this work has been mostly theoretical; few implementations exist to test the feasibility of these approaches on practical problems. Some exceptions are [26] and [6] where PVS is used to verify (part of) the steamboiler challenge problem [1].

On the other hand, algorithmic verification methods for hybrid systems, based on hybrid automata [2, 16], and implemented in the tool HyTech [18] have been successfully applied to many, relatively large practical examples, for example [20, 21]. However, HyTech is applicable only to rectangular hybrid automata, that is, systems with a finite control structure, in which the derivative of all continuous variables either is constant or lies in an interval bounded by constants. Although several ways have been identified to construct conservative rectangular approximations of systems that cannot be described by rectangular automata [17, 19], these steps may be informally justified and thus error prone. Although HyTech is able to do parametric analysis, due to the limitations of current polyhedra technology, it is usually restricted to systems with a few parameters unspecified; it expects fixed, explicit values for the rest of the parameters.

* This research was supported in part by the National Science Foundation under grant CCR-95-27927, the Defense Advanced Research Projects Agency under NASA grant NAG2-892, ARO under grant DAAH04-95-1-0317, ARO under MURI grant DAAH04-96-1-0341, and by Army contract DABT63-96-C-0096 (DARPA).

In general, algorithmic methods are preferable whenever they are applicable, because they are fully automatic. However, deductive methods are applicable to a larger class of systems, and, in general can handle systems with symbolic constants, parameterized systems, and nonlinear systems. The price of the generality of the deductive approach is the need for intermediate assertions and invariants and possibly interactive theorem proving.

In this paper we investigate the practical aspects of the deductive verification of hybrid systems by presenting a prototype implementation of a tool to assist in such verification. It is applicable to systems with infinite control structure and is not limited to rectangular hybrid systems. Our approach to the deductive verification of hybrid systems is based on the formalism of *phase transition systems*, introduced by Manna and Pnueli [22] as a model to describe hybrid systems. Phase transition systems are an extension of fair transition systems: *activities* are used to describe how continuous variables evolve over time, and a *progress condition* imposes constraints on the progress of time under various conditions.

Our tool is implemented as part of the STeP (Stanford Temporal Prover) verification system, an integrated toolset for verifying linear-time temporal properties of reactive systems. STeP's deductive methods include *verification rules* and *verification diagrams*. In [22] it is shown that phase transition systems define an associated transition system that has the same set of behaviours. In the associated transition system activities are translated into regular transitions parameterized by their duration. This correspondence makes STeP's deductive methods, originally developed for discrete systems, immediately applicable to hybrid systems.

STeP also provides tools for the automatic generation of invariants. STeP's invariant generation methods for discrete systems are described in [10], while [11] presents a method applicable to real-time systems. Here we adapt the method for real-time systems and propose an additional method that takes advantage of some properties of activities. We show that for some systems the invariants thus generated are sufficient to prove the properties of interest.

2 Preliminaries

2.1 Computational Model: Transition Systems

As the underlying computational model for verification we use *transition systems* [23]. A transition system $\Phi = \langle V, \Theta, \mathcal{T} \rangle$ consists of

- V : A finite set of typed *system variables*. A *state* is a type-consistent interpretation of the system variables. The set of all states is called the *state space*, and is designated by Σ . We say that a state s is a p -state if s satisfies p , written $s \models p$.
- Θ : The *initial condition*, a satisfiable assertion characterizing the initial states.
- \mathcal{T} : A finite set of *transitions*. Each transition $\tau \in \mathcal{T}$ is a function

$$\tau : \Sigma \mapsto 2^\Sigma$$

mapping each state $s \in \Sigma$ into a (possibly empty) set of τ -successor states, $\tau(s) \subseteq \Sigma$. Each transition τ is defined by a *transition relation* $\rho_\tau(V, V')$,

a first-order formula in which the unprimed variables refer to the values in the current state s , and the primed variables refer to the values in the next state s' . Transitions may be parameterized, thus simulating an infinite set of similar transitions.

Computations A computation of a transition system $\Phi = \langle V, \Theta, \mathcal{T} \rangle$ is an infinite sequence of states $\sigma : s_0, s_1, s_2, \dots$, such that

- *Initiation*: s_0 is initial, that is, $s_0 \models \Theta$.
- *Consecution*: For each $j = 0, 1, \dots$, s_{j+1} is a τ -successor of s_j , that is, $s_{j+1} \in \tau(s_j)$ for some $\tau \in \mathcal{T}$.

A computation prefix is a finite sequence of states that satisfies Initiation and Consecution.

2.2 System Description: Phase Transition Systems

Transition systems are not a very convenient formalisms to describe hybrid systems, because of the discrete nature of the transitions: transitions update the value of the variables in a discrete manner, rather than let the values of variables vary continuously over time. Therefore we use *phase transition systems* [22] to describe hybrid systems. The phase transition system presented here extends the one presented in [22] with *differential inclusions*.

A phase transition system (PTS) $\Psi = \langle V, \Theta, \mathcal{T}, \mathcal{A}, \Pi \rangle$ has the same components as a transition system plus two additional components that allow us to describe how continuous variables evolve over time. First, a PTS imposes some additional constraints on V , Θ , and \mathcal{T} :

- V : The set of system variables is partitioned into a set \mathcal{D} of *discrete variables*, which can be of any type, a set \mathcal{C} of *clock variables*, and a set \mathcal{I} of *continuous variables* (also known as *integrators*). All variables in \mathcal{C} and \mathcal{I} must be of type real. We assume that the set of clock variables includes a variable T , called the masterclock. The masterclock records the progress of global time.
- Θ : The initial condition must satisfy $\Theta \rightarrow T = 0$.
- \mathcal{T} : The transitions in \mathcal{T} are considered discrete and are assumed to happen instantaneously; therefore we require that no transition modify the master clock, that is, for every transition $\tau \in \mathcal{T}$ we require:

$$\rho_\tau(V, V') \rightarrow T' = T.$$

The new components in a PTS are

- \mathcal{A} : A finite set of *activities*. Each activity $\alpha \in \mathcal{A}$ is described by an *activity relation*:

$$\rho_\alpha : p_\alpha \rightarrow \dot{\mathcal{I}}_d^\alpha = F^\alpha(V) \wedge \dot{\mathcal{I}}_b^\alpha \in [F_l^\alpha(V), F_u^\alpha(V)]$$

where p_α , called the *activation condition*, is a predicate over \mathcal{D} , and $\mathcal{I}_d^\alpha \cup \mathcal{I}_b^\alpha = \mathcal{I}$, the set of integrators. \mathcal{I}_d^α is the set of variables for which the derivative is fully specified in α , while the derivatives of the variables in \mathcal{I}_b^α are specified by differential inclusions.

Activity α is said to be *active* in state s if its activation condition p_α holds on s . The formula $\dot{I}_d^\alpha = F^\alpha(V)$ stands for

$$\dot{x}_i = F_i^\alpha(V), \quad \text{for } i = 1, \dots, m$$

and the formula $\dot{I}_b^\alpha \in [F_l^\alpha(V), F_u^\alpha(V)]$ stands for the *differential inclusion*

$$\dot{x}_i \in [F_{i,l}^\alpha(V), F_{i,u}^\alpha(V)], \quad \text{for } i = m+1, \dots, n$$

where $\{x_1, \dots, x_n\} = \mathcal{I}$. The functions F_i^α specify how the continuous variables change over time while the system is in a p_α -state; the functions $F_{i,l}^\alpha(V), F_{i,u}^\alpha(V)$ are a lower and upper bound on the derivative of x_i . Each activity must specify an evolution constraint on each continuous variable. We assume that the integral of each F^α is well-defined and we require that F^α does not depend on variables specified by a differential inclusion.

Activities should be *time-invariant*, that is, F_i^α cannot explicitly refer to time elapsed since the start of the activity. This condition does not reduce the expressiveness, but may require the introduction of additional variables. For example, to specify that a variable x varies according to the square root of time in some activity, we cannot say $\dot{x} = \sqrt{t}$, but we have to say $\dot{x} = \sqrt{y}, \dot{y} = 1$. The reason for this condition is to make sure that the effects of two consecutive τ_α steps are the same as the effects of one single τ_α step with duration the sum of the two steps.

To ensure that the phase transition system is *time-deterministic* we require that the activities' activation conditions are mutually exclusive and exhaustive, that is $p_{\alpha_1} \rightarrow \neg p_{\alpha_2}$ for $\alpha_1 \neq \alpha_2$, and $\bigvee_{\alpha \in \mathcal{A}} p_\alpha$ must hold on the reachable states.

- Π : The *time-progress condition*. An assertion over V used to specify a global restriction over the progress of time.

Associated Transition System With each activity $\alpha \in \mathcal{A}$ we associate a parameterized transition $\tau[\Delta]$, which represents an infinite set of transitions, one for each possible interval duration Δ . We refer to these transitions as time-step transitions: these are the only transitions that can advance global time. If α has activity relation

$$p_\alpha \rightarrow \dot{I}_d^\alpha = F^\alpha(V) \wedge \dot{I}_b^\alpha \in [F_l^\alpha(V), F_u^\alpha(V)]$$

the transition relation of $\tau_\alpha[\Delta]$ is given by

$$\rho_{\tau_\alpha}[\Delta] : \left(\begin{array}{c} \Delta > 0 \wedge p_\alpha \wedge \mathcal{D}' = \mathcal{D} \wedge \mathcal{C}' = \mathcal{C} + \Delta \\ \wedge \\ \dot{I}_d^{\alpha'} = \dot{I}_d^\alpha + G^\alpha(\Delta) \\ \wedge \\ \dot{I}_b^\alpha + G_l^\alpha(\Delta) \leq \dot{I}_b^{\alpha'} \wedge \dot{I}_b^{\alpha'} \leq \dot{I}_b^\alpha + G_u^\alpha(\Delta) \\ \wedge \\ \forall E \in \mathbb{R} \forall \delta \in (0, \Delta]. \left(\begin{array}{c} \dot{I}_b^\alpha + G_l^\alpha(\delta) \leq E \wedge E \leq \dot{I}_b^\alpha + G_u^\alpha(\delta) \\ \rightarrow \\ \Pi(\mathcal{D}, \mathcal{C} + \delta, \dot{I}_d^\alpha + G^\alpha(\delta), E) \end{array} \right) \end{array} \right)$$

where

$$G^\alpha(\delta) = \int_0^\delta F^\alpha dt, \quad G_l^\alpha(\delta) = \int_0^\delta F_l^\alpha dt, \quad G_u^\alpha(\delta) = \int_0^\delta F_u^\alpha dt$$

and $\Pi(\mathcal{D}, \mathcal{C}, \mathcal{I}_a^\alpha, \mathcal{I}_b^\alpha)$ is the progress condition.

In words, each time-step transition $\tau_\alpha[\Delta]$ is a transition that is enabled if it has a positive time duration Δ , its activity condition p_α holds, and the progress condition holds throughout the interval $(0, \Delta]$ for all values of the derivatives of the variables in \mathcal{I}_b^α . It is assumed that during this interval all continuous variables evolve according to the derivatives specified in the activity relation, all clocks increase uniformly with time, and all discrete variables stay the same. If the transition is taken, the values of the variables in the successor state(s) are constrained by the primed expressions in the transition relation.

The progress condition is similar to the *tcp* predicate introduced in [24]: it constrains the time that the system can reside in a particular configuration.

The phase transition system $\Psi = \langle V, \Theta, \mathcal{T}, \mathcal{A}, \Pi \rangle$ defines the associated transition system $\Phi = \langle V, \Theta, \mathcal{T}_H \rangle$, where

$$\mathcal{T}_H = \mathcal{T} \cup \mathcal{T}_A, \quad \text{where } \mathcal{T}_A = \{\tau_\alpha[\Delta] \mid \alpha \in \mathcal{A}, \Delta \in \mathbb{R}^+\}$$

Computations A computation of a PTS Ψ is an infinite sequence of states $\sigma : s_0, s_1, s_2, \dots$, such that

- σ is a computation of Φ , where Φ is the associated transition system defined by Ψ , and
- *Time Divergence*: the value of the masterclock T grows beyond any bound, that is, the sequence $s_0[T], s_1[T], \dots$ grows beyond any bound.

A hybrid system is called *non-Zeno* if every finite sequence of states that is a computation prefix of the associated transition system can be extended into a computation. In this paper we restrict ourselves to non-Zeno systems.

2.3 Verification of safety properties

A *safety property* is a property expressible by a formula of the form $\Box p$, for a past temporal formula p (see [23] for definitions of past formula and the semantics of \Box). This includes *invariances*, where p is an assertion. In this case the formula states that p should be true in every accessible state of the system.

Because of the possibility to associate a transition system with a phase transition system, many verification rules presented in [23] can be reused for the verification of phase transition systems. In this paper we will use the *invariance rule* *INV*, shown in Figure 1, to prove some properties of hybrid systems. These rules reduce the system validity of a temporal formula to the general validity of a set of first-order verification conditions. In the rules $\{\varphi\} \tau \{\psi\}$ stands for the formula

$$(\varphi(V) \wedge \rho_\tau(V, V')) \rightarrow \psi(V'), \quad \forall \Delta. (\varphi(V) \wedge \rho_\tau[\Delta](V, V')) \rightarrow \psi(V')$$

for a regular and a parameterized transition τ , respectively.

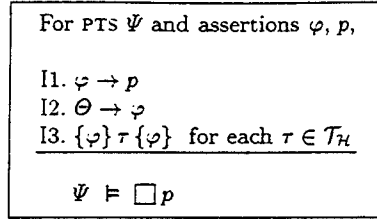


Fig. 1. Invariance rule INV

We say that an assertion p is *inductive* for a hybrid system Ψ if $\Box p$ can be proved using rule INV with φ equal to p (that is, p holds initially and is preserved by every transition). If these verification conditions can be proved assuming a set of properties S , we say that p is *inductive relative to S* .

3 STeP

The Stanford Temporal Prover, STeP, is a tool for the deductive and algorithmic verification of reactive systems [8, 9, 11].

STeP implements *verification rules* and *verification diagrams* for deductive verification. A collection of decision procedures for built-in theories, including integers, reals, datatypes and equality is combined with propositional and first-order reasoning to simplify verification conditions, proving many of them automatically. For those that cannot be established automatically, an interactive Gentzen-style theorem prover is available. Features such as parameterization and transitions originating from activities introduce quantifiers in verification conditions. Fortunately, the required quantifier instantiations are often “obvious” in that they use instances that can be provided by the decision procedures themselves. Accordingly, an integration of first-order reasoning and decision procedures was developed that can automatically discharge many verification conditions that would otherwise require the use of the interactive prover [12].

To enable symbolic manipulation of first-order formulas in the theory of real closed fields, we are planning to integrate STeP with REDLOG [13], a package that forms a front-end to the computer algebra system REDUCE [14]. Some of the verification conditions generated by the case studies reported in this paper were proved automatically by a version of REDLOG made available on the web.

4 Generation of Invariants

STeP provides tools for automatic generation of invariants based on static analysis of transition systems for reactive systems [10] and real-time systems [11]. These invariants are invaluable as auxiliary properties in deductive verification. We will describe two techniques for automatically generating invariants for hybrid systems.

For a PTS $\Psi = \langle V, \Theta, \mathcal{T}_D, \mathcal{A}, \Pi \rangle$, and associated transition system $\Phi = \langle V, \Theta, \mathcal{T}_H \rangle$, define

$$Post_D(X) = \bigvee_{\tau \in \mathcal{T}_D} post(\tau, X), \quad Post_A(X) = \bigvee_{\tau \in \mathcal{T}_H - \mathcal{T}_D} post(\tau, X)$$

where

$$post(\tau, X) = \exists V^0. X(V^0) \wedge \rho_\tau(V^0, V)$$

and for a parameterized transition $\tau[\Delta]$

$$post(\tau, X) = \exists \Delta, V^0. X(V^0) \wedge \rho_\tau[\Delta](V^0, V)$$

Thus, $Post_D(X)$ characterizes all the states that can be reached from a state satisfying X by a discrete transition, and $Post_A(X)$ characterizes all the states that can be reached from a state satisfying X by a time-step transition.

Invariant 1 The first invariant is similar to that described in [11] for real-time systems.

$$Inv_1 : \Theta \vee Post_D(true) \vee Post_A(true)$$

Inv_1 characterizes the set of states that is either an initial state or can be reached by either a discrete transition or a time-step transition, starting from anywhere in the state space. It is not hard to see that Inv_1 is an invariant of Ψ .

As we may want to apply an invariant to every verification condition, it is desirable to minimize the number of quantifiers it contains. In STeP the existential quantifiers generated for the discrete transitions are used with universal force when appearing in assumptions and are mostly eliminated by STeP's simplifier. Similar to [11] we can approximate $Post_A(true)$ by the progress condition Π , since

$$post(\tau_\alpha, true) = \exists \Delta, \mathcal{D}^0, \mathcal{C}^0, \mathcal{I}^0. \rho_{\tau_\alpha}[\Delta](V^0, V)$$

is equivalent to

$$p_\alpha \wedge \exists \Delta \left(\begin{array}{c} \Delta > 0 \\ \wedge \\ I_b^{\alpha 0} + G_i^\alpha(\Delta) \leq I_b^\alpha \wedge I_b^\alpha \leq I_b^{\alpha 0} + G_u^\alpha(\Delta) \\ \wedge \\ \exists I_b^{\alpha 0} \left(\begin{array}{c} I_b^{\alpha 0} + G_i^\alpha(\delta) \leq E \wedge E \leq I_b^{\alpha 0} + G_u^\alpha(\delta) \\ \rightarrow \\ \Pi(\mathcal{D}, \mathcal{C} + \delta - \Delta, I_d^\alpha + G^\alpha(\delta) - G^\alpha(\Delta), E) \end{array} \right) \end{array} \right)$$

by taking $\mathcal{D}^0 = \mathcal{D}$, $\mathcal{C}^0 = \mathcal{C} - \Delta$ and $\mathcal{I}_d^{\alpha 0} = I_d^\alpha - G^\alpha(\Delta)$, which in turn implies

$$p_\alpha \wedge \Pi(\mathcal{D}, \mathcal{C}, I_d^\alpha, I_b^\alpha)$$

by taking $\delta = \Delta$ and $E = I_b^\alpha$, and thus we have

$$Post_A \rightarrow \Pi(\mathcal{D}, \mathcal{C}, I_d^\alpha, I_b^\alpha)$$

as required.

In [7] and [25] a similar method is used for the generation of invariants for untimed programs and hardware respectively.

Invariant 2 The second invariant takes advantage of the time-invariance property of activities. Time invariance ensures that the possible effects of taking two successive τ_α transitions of duration Δ_1 and Δ_2 are the same as taking one τ_α transition of duration $\Delta_1 + \Delta_2$, that is

$$\rho_{\tau_\alpha}[\Delta_1] \circ \rho_{\tau_\alpha}[\Delta_2] = \rho_{\tau_\alpha}[\Delta_1 + \Delta_2]$$

Based on this property we can claim

Claim 1 *Given a phase transition system $\Phi = \langle V, \Theta, \mathcal{T}, \mathcal{A}, \Pi \rangle$, the following is an invariant of Φ :*

$$Inv_2 : \Theta \vee Post_D(true) \vee Post_A(Post_D(true) \vee \Theta)$$

Justification: Assume, by contradiction, that there is some state s that is accessible in a computation of Ψ , but does not satisfy Inv_2 . Clearly s cannot be an initial state or the result of a discrete transition, so it must be the final state of a time-step transition, $\tau_\alpha[\Delta]$. Let s^0 be the state from which τ_α was taken. Clearly s^0 cannot be the result of a discrete transition, or an initial state, so it must, like s , be the final state of a time-step transition, τ_{α_1} . However, τ_{α_1} , where $\alpha_1 \neq \alpha$ cannot be followed immediately by τ_α , by the requirement that $p_{\alpha_1} \wedge p_\alpha$ be unsatisfiable, that the activation conditions only depend on discrete variables, and that a time-step transition cannot modify any discrete variables; two distinct time-step transitions always have to be separated by a discrete transition. Thus s^0 must be the final state of another time-step transition $\tau_\alpha[\Delta_1]$. However, by time invariance, the effect of $\tau_\alpha[\Delta_1]$ followed by $\tau_\alpha[\Delta]$ is the same as taking the single time-step transition $\tau_\alpha[\Delta_1 + \Delta]$; repeating the same argument for the starting state of $\tau_\alpha[\Delta_1]$ we can conclude, by induction, that $\tau_\alpha[\Delta]$ cannot be preceded by another τ_α time-step transition.

In the following section we will see that this invariant is strong enough to prove the safety property of the water-level monitor.

5 Example

We verified several (symbolic versions) of the case studies reported in the HyTech literature. Translation from a hybrid automaton [16] to a phase transition system is straightforward. Given a hybrid automaton $\mathcal{H} = \langle X, (V, E), init, inv, flow, jump \rangle$ where X is a set of variables, (V, E) a set of nodes and edges, $init$ a mapping from nodes to assertions denoting the initial condition, inv a mapping from nodes to assertions denoting node invariants, $flow$ a mapping from nodes to relations over $X \cup \dot{X}$ specifying the derivatives of the continuous variables, and $jump$ a mapping from edges to relations over $X \cup X'$ denoting the discrete transitions, the corresponding phase transition system is $\Psi = \langle X \cup \{s\}, \Theta, \mathcal{T}, \mathcal{A}, \Pi \rangle$, where s is a new (discrete) variable with domain V ,

$$\begin{aligned} \Theta &= \bigvee_{v \in V} (s = v \wedge init(v)), \\ \mathcal{T} &= \{ \tau \mid \rho_\tau = jump(e) \wedge e \in E \}, \end{aligned}$$

$$A = \{\alpha \mid \rho_\alpha = (s = v \rightarrow \text{flow}(v)) \wedge v \in V\},$$

$$I = \bigwedge_{v \in V} s = v \rightarrow \text{inv}(v) .$$

5.1 Water-level monitor

To illustrate our methods we will describe the verification of the water-level monitor system shown in Figure 2, taken from [3]; its description as a hybrid

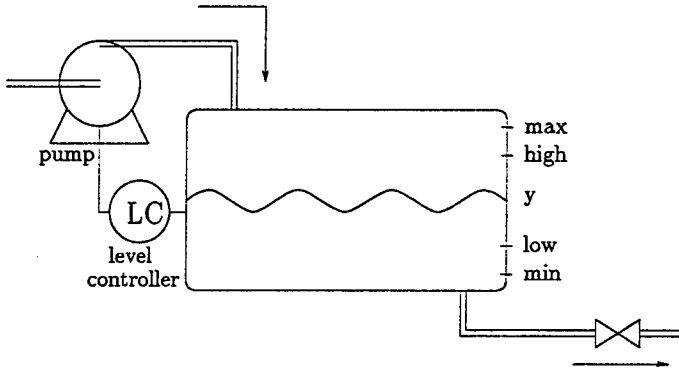


Fig. 2. Water-level monitor system

automaton is shown in Figure 3, and its description as a phase transition system, as entered in STeP is shown in Figure 4. The system consists of a watertank

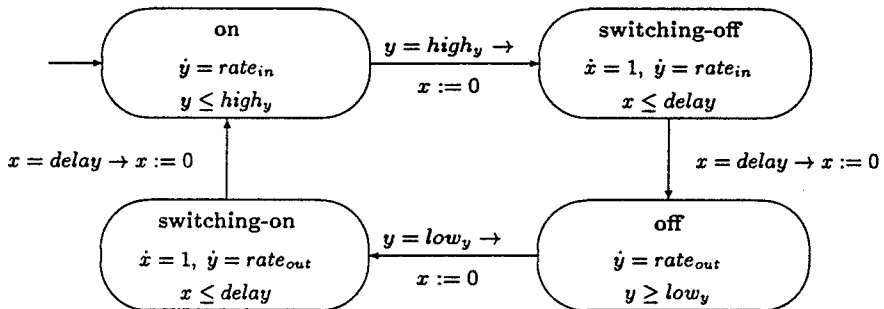


Fig. 3. Water level monitor - hybrid automaton

Hybrid Transition System Waterlevel Controller

```

type valveStates = {on, switching_off, off, switching_on}

in min_y, max_y : real where min_y <= max_y
in low_y: real where low_y >= min_y
in high_y: real where high_y <= max_y /\ high_y > low_y
in delay: real where delay > 0
in rate_in, rate_out: real where rate_in > 0 /\ rate_out < 0

local s : valveStates where s = on
clock x where x = 0
continuous y where y = low_y

Progress
  (s = on --> y <= high_y) /\
  (s = switching_off --> x <= delay) /\
  (s = off --> y >= low_y) /\
  (s = switching_on --> x <= delay)

Transition switch_off:
  enable s = on /\ y = high_y
  assign s := switching_off, x := 0

Transition isoff:
  enable s = switching_off /\ x = delay
  assign s := off, x := 0

Transition switch_on:
  enable s = off /\ y = low_y
  assign s := switching_on, x := 0

Transition ison:
  enable s = switching_on /\ x = delay
  assign s := on, x := 0

Activity Aon:
  enable s = on \/ s = switching_off
  assign Deriv(y) := rate_in

Activity Aoff:
  enable s = off \/ s = switching_on
  assign Deriv(y) := rate_out

```

Fig. 4. The Waterlevel Controller phase transition system

that supplies water to a customer at a constant rate. The level, y , in the tank is controlled by a controller, which observes the level via a level sensor. When the level drops below low_y , the controller starts a pump to refill the tank, and when the level rises above $high_y$ the pump is turned off again. When the pump is on, the level rises with rate $rate_{in}$, when the pump is off, the level drops with rate $rate_{out}$. However, there is a time delay of $delay$ seconds between the time the controller sends the signal to the pump and the time the flow is established or the pump is stopped

The property we want to prove about this system is that the level stays between a lower and upper limit, expressed by the linear-time temporal logic formula

$$safe : \Box (min_y \leq y \wedge y \leq max_y)$$

assuming there is sufficient margin between max_y and $high_y$, and between min_y and low_y , expressed by the axioms

$$\begin{aligned} max_y &\geq high_y + rate_{in} * delay \\ min_y &\leq low_y + rate_{out} * delay \end{aligned}$$

Not surprisingly, the property *safe* is not inductive, that is, after application of rule INV, taking $\varphi = p$, all first-order verification conditions simplify to true automatically except those for *Aon* and *Aoff*, which in fact are not valid. Rather than trying to strengthen the property to make it inductive, we generate the invariants described in Section 4.

We first generate $Post_D(true)$, which results (after simplification) in

$$Post_D(true) : x = 0 \wedge \left(\begin{array}{c} s = \text{on} \vee s = \text{off} \\ \vee \\ low_y = y \wedge s = \text{switching-on} \\ \vee \\ high_y = y \wedge s = \text{switching-off} \end{array} \right)$$

and we use this to generate $Post_A(Post_D \vee \Theta)$, resulting in

$$Post_A(Post_D \vee \Theta) : \left(\begin{array}{c} x > 0 \\ \wedge \\ (s = \text{off} \rightarrow low_y \leq y) \\ \wedge \\ (s = \text{on} \rightarrow y \leq high_y) \\ \wedge \\ ((s = \text{switching-on} \vee s = \text{switching-off}) \rightarrow x \leq delay) \\ \wedge \\ \left(\begin{array}{c} s = \text{off} \vee s = \text{on} \\ \vee \\ s = \text{switching-on} \wedge y = low_y + rate_{out} \cdot x \\ \vee \\ s = \text{switching-off} \wedge y = high_y + rate_{in} \cdot x \end{array} \right) \end{array} \right)$$

Taking the disjunction, we obtain the invariants we need to make the property *safe* inductive:

$$\begin{aligned} s = \text{switching-on} &\rightarrow y = \text{low}_y + \text{rate}_{\text{out}} \cdot x \wedge x \leq \text{delay} \\ s = \text{switching-off} &\rightarrow y = \text{high}_y + \text{rate}_{\text{in}} \cdot x \wedge x \leq \text{delay} \\ x &\geq 0 \end{aligned}$$

With these invariants the two remaining verification conditions simplify to true, where some of the non-linear clauses were proved by REDLOG [13].

Note that the system verified here cannot be verified by the current version of HyTech due to the use of symbolic constants for rate constants which results in non-linear terms.

5.2 Other Systems Verified

Other systems verified using STeP include the temperature controller [2], the railroad crossing, the three versions of the nuclear reactor (clock translation, linear approximation, and rectangular approximation) [5], and the cat and mouse example [22]. In most of these systems the automatic invariant generator generated some of the required invariants, but the user had to supply additional invariants to make the main safety property inductive. Verification of the above systems with symbolic constants instantiated with numbers were mostly automatic, apart from providing some invariants not provided by the automatic invariant generator. Verification of these systems with symbolic constants generally required some, usually trivial, user guidance in the interactive theorem prover. More examples of hybrid systems verified with STeP will appear on the STeP webpage: <http://rodin.stanford.edu/>.

6 Conclusion

We demonstrated the feasibility of computer-aided deductive verification of hybrid systems. We verified with STeP symbolic versions of (admittedly small) examples previously verified by HyTech. The verification of the symbolic versions usually required some user interaction, the verification of the instantiated systems (that is, the systems verified by HyTech) was mostly automatic (apart from providing some invariants). Currently the main limitation is the lack of decision procedures for real arithmetic, which makes it necessary to prove some, mathematically trivial, first-order verification conditions interactively, which is tedious. Hopefully this problem will be ameliorated with the integration of REDLOG.

Considering that the current implementation is still rather limited, our preliminary results suggest a high potential for deductive methods for the verification of hybrid systems.

Acknowledgements: We thank Nikolaj Bjørner and Tomás Uribe for their feedback and comments.

References

1. ABRIAL, J. R., BÖRGER, E., AND LANGMAACK, H., Eds. *Formal Methods for Industrial Applications*, vol. 1165 of *LNCS*. Springer-Verlag, 1996.
2. ALUR, R., COURCOUBETIS, C., HALBWACHS, N., HENZINGER, T. A., HO, P.-H., NICOLLIN, X., OLIVERO, A., SIFAKIS, J., AND YOVINE, S. The algorithmic analysis of hybrid systems. *Theoretical Computer Science* 138, 1 (1995), 3–34.
3. ALUR, R., COURCOUBETIS, C., HENZINGER, T. A., AND HO, P.-H. Hybrid automata: An algorithmic approach to the specification and analysis of hybrid systems. In Grossman et al. [15], pp. 209–229.
4. ALUR, R., AND HENZINGER, T. A., Eds. *Proc. 8th Intl. Conference on Computer Aided Verification* (July 1996), vol. 1102 of *LNCS*, Springer-Verlag.
5. ALUR, R., HENZINGER, T. A., AND HO, P. Automatic symbolic verification of embedded systems. *IEEE Trans. Software Engin.* 22, 3 (Mar. 1996), 181–201.
6. ARCHER, M., AND HEITMEYER, C. Verifying hybrid systems modeled as timed automata: A case study. In *Proc. 1st Intl. Workshop Hybrid and Real-time Systems (HART)* (1997), O. Maler, Ed., vol. 1201 of *LNCS*, Springer-Verlag.
7. BENSALÉM, S., LAKHNECH, Y., AND SAIDI, H. Powerful Techniques for the Automatic Generation of Invariants. In Alur and Henzinger [4], pp. 323–335.
8. BJØRNER, N. S., BROWNE, A., CHANG, E. S., COLÓN, M., KAPUR, A., MANNA, Z., SIPMA, H. B., AND URIBE, T. E. STeP: Deductive-algorithmic verification of reactive and real-time systems. In Alur and Henzinger [4], pp. 415–418.
9. BJØRNER, N. S., BROWNE, A., CHANG, E. S., COLÓN, M., KAPUR, A., MANNA, Z., SIPMA, H. B., AND URIBE, T. E. STeP: The Stanford Temporal Prover, User's Manual. Tech. Rep. STAN-CS-TR-95-1562, Computer Science Department, Stanford University, Nov. 1995.
10. BJØRNER, N. S., BROWNE, A., AND MANNA, Z. Automatic generation of invariants and intermediate assertions. *Theoretical Computer Science* 173, 1 (Feb. 1997), 49–87. Preliminary version appeared in 1st Intl. Conf. on Principles and Practice of Constraint Programming, vol. 976 of *LNCS*, pp. 589–623, Springer-Verlag, 1995.
11. BJØRNER, N. S., MANNA, Z., SIPMA, H. B., AND URIBE, T. E. Deductive verification of real-time systems using STeP. In *4th Intl. AMAST Workshop on Real-Time Systems* (May 1997), vol. 1231 of *LNCS*, Springer-Verlag, pp. 22–43.
12. BJØRNER, N. S., STICKEL, M. E., AND URIBE, T. E. A practical integration of first-order reasoning and decision procedures. In *Proc. of the 14th Intl. Conference on Automated Deduction* (July 1997), vol. 1249 of *LNCS*, Springer-Verlag, pp. 101–115.
13. DOLZMANN, A., AND STURM, T. REDLOG: Computer algebra meets computer logic. *ACM SIGSAM Bulletin* 31, 2 (June 1997), 2–9.
14. FÜR INFORMATIONSTECHNIK BERLIN, K. Z. Z. REDUCE symbolic math system. <http://www.zib.de/Symbolik/reduce/>, 1995.
15. GROSSMAN, R. L., NERODE, A., RAVN, A. P., AND RISCHEL, H., Eds. *Hybrid Systems* (1993), vol. 736 of *LNCS*, Springer-Verlag.
16. HENZINGER, T. A. The theory of hybrid automata. In *Proc. 11th IEEE Symp. Logic in Comp. Sci.* (1996), IEEE Computer Society Press, pp. 278–292.
17. HENZINGER, T. A., AND HO, P. Algorithmic analysis of nonlinear hybrid systems. In Wolper [27], pp. 225–238.
18. HENZINGER, T. A., HO, P., AND WONG-TOI, H. A user guide to HYTECH. In *TACAS 95: First Intl. Workshop on Tools and Algorithms for the Construction and Analysis of Systems* (1995), E. Brinksma, W. Cleaveland, K. Larsen, T. Margaria, and B. Steffen, Eds., vol. 1019 of *LNCS*, Springer-Verlag, pp. 41–71.

19. HENZINGER, T. A., AND WONG-TOI, H. Linear phase-portrait approximations for nonlinear hybrid systems. In *Hybrid Systems III* (1996), R. Alur, T. A. Henzinger, and E. D. Sontag, Eds., vol. 1066 of *LNCS*, Springer-Verlag, pp. 377-388.
20. HENZINGER, T. A., AND WONG-TOI, H. Using HYTECH to synthesize control parameters for a steam boiler. In Abrial et al. [1].
21. HO, P.-H., AND WONG-TOI, H. Automated analysis of an audio control protocol. In Wolper [27], pp. 381-394.
22. MANNA, Z., AND PNUELI, A. Clocked transition systems. In *Proc. of the Intl. Logic and Software Engineering Workshop* (Aug. 1995). Beijing, China.
23. MANNA, Z., AND PNUELI, A. *Temporal Verification of Reactive Systems: Safety*. Springer-Verlag, New York, 1995.
24. NICOLLIN, X., OLIVERO, A., SIFAKIS, J., AND YOVINE, S. An approach to the description and analysis of hybrid systems. In Grossman et al. [15], pp. 149-178.
25. SU, J. X., DILL, D. L., AND BARRETT, C. W. Automatic generation of invariants for processor verification. In *1st Intl. Conf. on Formal Methods in Computer-Aided Design* (Nov. 1996), vol. 1166 of *LNCS*, Springer-Verlag, pp. 377-388.
26. VITT, J., AND HOOMAN, J. Assertion specification and verification using PVS of the steam boiler control system. In Abrial et al. [1], pp. 453-472.
27. WOLPER, P., Ed. *Proc. 7th Intl. Conference on Computer Aided Verification* (July 1995), vol. 939 of *LNCS*.

Reduction and Decomposition of Differential Automata: Theory and Applications

Alexey S. Matveev and Andrey V. Savkin

Department of Electrical and Electronic Engineering,
The University of Western Australia,
Nedlands, Perth, WA 6009, Australia

e-mail: savkin@ee.uwa.edu.au, fax: +618 93801065; phone: +618 93801621

Abstract. The paper considers an important class of hybrid dynamical systems called differential automata. A differential automaton is said to be reducible if its dynamics can be described by some discrete automaton with a finite number of states. Our main results show that, under certain general assumptions, any differential automaton is reducible. Furthermore, we prove that any reducible differential automaton can be represented as a union of a finite number of differential automata with simple cyclic dynamics. Moreover, we show that the differential automaton has a periodic trajectory corresponding to each of this cyclic automata. For planar differential automata, we derive an analog of the classic Poincaré-Bendixon theorem.

1 Introduction

Hybrid dynamical systems (**HDS**) have attracted considerable attention in recent years (see, e.g., [1, 2, 3]). In general, **HDS** are those that combine continuous and discrete behavior and involve, thereby, both continuous and discrete state variables. In many cases (but not always), such systems operate as follows. While the discrete state remains constant, the continuous one obeys a definite dynamical law. Transition to another discrete state implies a change of this law. In its turn, the discrete state evolves as soon as a certain event occurs with both the evolution and the event depending on the continuous state.

A typical hybrid system is a logical discrete-event decision-making system interacting with a continuous time process. A simple example is a home climate-control system. Due to its on-off nature, the thermostat is modelled as a discrete-event system, whereas the furnace or air-conditioner are modelled as continuous-time systems. Some other examples concern vehicle transmission systems and stepper motors, computer disk drivers, robotics systems, higher-level flexible manufacturing systems, intelligent vehicle/highway systems, sea/air traffic management systems as well as various systems with relays, switches, and hysteresis (see, e.g., [1-9]).

Numerous attempts have been made recently to develop a general approach to analysis and design of hybrid control systems. (See, e.g., [1, 2, 3] and the literature therein.) One of the ideas employed was that of algebraic reduction to

a finite-state automaton [4, 11, 12]. It proceeds from the fact that the relevant dynamics of certain discrete events associated with the system can be, in some cases, described with such an automaton. Roughly speaking, this means that the discrete component of the dynamics can be studied independently of the continuous one. In its turn, this is a key to the analysis of the dynamics in full. The above approach was so far justified for and most fruitfully applied to relatively simple **HDS** called timed automata [11, 12] as well as special flow models of manufacturing systems (see, e.g., the switched server system example in [10]).

In this paper, the problem of the algebraic reducibility is considered for a quite general model of **HDS** called a *differential automaton* (**DA**). This model was introduced in [13] to describe various control systems with hysteresis. Under more general assumptions, it covers a fairly larger variety of **HDS** including those with sliding-mode phenomena [15, 16]. **DA** is studied in a bounded connected invariant domain. The main goal of the paper is to demonstrate that the algebraic reducibility follows directly from several general dynamical properties of the system such as well-posedness, absence of singular points on the surfaces where the discrete states changes and some other ones. More precisely, whenever these properties hold, the discrete state untimed behavior is governed by a finite state automaton.

The algebraic reducibility not only means that the dynamical behavior of the discrete state is quite simple but also often constitutes a breakthrough to analysis of the dynamics in full. To illustrate this, it is shown in Section 4 that, under an additional assumption, the above invariant domain necessarily contains a trajectory, which is periodic in both the continuous and the discrete states. Another example is given in Section 5 where an analog of the classic Poincaré-Bendixon theorem [22, p.295] is established for planar **DA**. Though the focus on two dimensional **HDS** is a severe restriction, they have attracted considerable attention recently. A very special example of such system (i.e., the so-called three buffer switched arrival/server system) was investigated in [10, 17, 18, 19]. The emphasis was to distinguish between the cases when the dynamics is chaotic and, respectively, when any trajectory converges to a limit cycle. In [20], an affine **HDS** was investigated. Such system consists of a partition of the Euclidian space into a finite set of polyhedral regions. Within each region, the dynamics is defined by a constant vector field. Discrete transitions occur only on the boundaries between regions. Certain reachability problem was studied with the main result being a decision procedure for two-dimensional systems. In [21], a generic class of planar **HDS** was considered. It was shown how the complexity of such system can be reduced to a one-dimensional transformation by inducing the system onto a set of curves and certain properties of the induced system was investigated. In Section 5, we establish an analog of the Poincaré-Bendixon theorem for a quite general model of a discontinuous planar **HDS**. More precisely, it is shown that some general dynamical properties of the system imply that the dynamics is nonchaotic.

The body of the paper is organized as follows. Section 2 contains basic as-

sumptions and definitions. A necessary and sufficient criterion for **DA** to be well-posed is also given here. In Section 3, we present the main results. In particular, we show that **DA** can be decomposed into a finite number of **DA** such that, for any of them, the dynamics of the discrete state is governed by a simple finite state automaton. In Section 4, we establish existence of periodic trajectories. Section 5 contains an analog of the classic Poincaré-Bendixon theorem.

The following notations are adopted throughout the paper. $\{p_1, \dots, p_s\}$ is the set consisting of the elements listed. \mathbb{R}^n is equipped with the Euclidean norm denoted as $|\cdot|$. Let $K \subset \mathbb{R}^n$ and $E \subset K$ be given. The symbol \overline{E}^K stands for the relative closure of E in K , i.e., \overline{E}^K is the set of all points $a \in K$ that can be approached $a = \lim_{i \rightarrow \infty} a_i$ by a sequence $\{a_i\} \subset E$. Likewise, $\text{int}_K E$ is the relative interior of E in K , i.e., $\text{int}_K E$ consists of all points $a \in E$ such that, along with a , the set E contains all neighboring points $a' \in K$: $a' \in K \& |a' - a| < \varepsilon \Rightarrow a' \in E$ for some $\varepsilon > 0$. The symbol $\partial_K E$ denotes the relative boundary of E in K , i.e., the collection of all points from \overline{E}^K that do not belong to $\text{int}_K E$. If $K = \mathbb{R}^n$, the index K is dropped in the notations $\text{int}_K E, \overline{E}^K, \partial_K E$.

2 Basic assumptions and definitions

Consider the following model of **HDS** called a *differential automaton* [13]

$$\dot{x}(t) = f[x(t), q(t)], \quad (2.1)$$

$$q(t+0) = \varphi[x(t), q(t)]. \quad (2.2)$$

Here $x(t) \in \mathbb{R}^n$ and $q(t) \in Q$ are, respectively, the continuous-valued and the discrete states, Q is a finite set of discrete states, and $f(\cdot) : \mathbb{R}^n \times Q \rightarrow \mathbb{R}^n, \varphi : \mathbb{R}^n \times Q \rightarrow Q$ are given functions. Assume that this system satisfies the following assumptions **A.1)**-**A.5)**.

A.1) For each $p \in Q$, the function $f(\cdot, p) : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is continuously differentiable. Any solution $x(\cdot)$ of the equation $\dot{x}(t) = f[x(t), p], t \geq 0$ can be extended on the interval $[0, +\infty)$ and $|x(t)| \rightarrow \infty$ as $t \rightarrow \infty$. For any $p, q \in Q, p \neq q$, the set $T_{p \rightarrow q} := \{x : \varphi(x, p) = q\}$ is closed.

This, in particular, implies that the set

$$H_p := \{x : \varphi(x, p) = p\} = \left\{ x \in \mathbb{R}^n : x \in \bigcup_{q \neq p} T_{p \rightarrow q} \right\} \quad (2.3)$$

is open for all $p \in Q$. It also means that the dynamics within a given discrete state is simple. (We assume this for the sake of simplicity.) The next assumption resembles the nonsingularity one from the theory of differential equations with discontinuous righthand sides [14]. This assumption stipulates that the vector fields are not tangent to the surfaces of discontinuity points. So far as the

boundary $\partial T_{p \rightarrow q}$ is not required to be smooth now, we need a more complicated formulation.

For any $a, h \in \mathbb{R}^n, r, \eta > 0$, denote by $k_r^\eta(a, h)$ the cone with the vertex a and the axis h defined as $k_r^\eta(a, h) := \{x \in \mathbb{R}^n : x = a + t(h + \Delta) \text{ for some } 0 < t \leq \eta, |\Delta| \leq r\}$ and say that, at the point a , the vector h looks at a set $G \subset \mathbb{R}^n$ if $k_r^\eta(a, h) \subset G$ for some $\eta, r > 0$. By this definition, the vector h does so whenever a is an interior point of G .

A.2) For any $p, q \in Q, p \neq q$, and $a \in \partial T_{p \rightarrow q}$, the vector $f(a, p)$ looks at $T_{p \rightarrow q}$ whenever it does not look at H_p .

If the boundary $\partial T_{p \rightarrow q}$ is a C^1 -surface, this assumption merely means that the vector $f(a, p)$ is not tangent to $\partial T_{p \rightarrow q}$ at the point a .

There are several definitions of the solution of the system (2.1), (2.2) in the literature (see, e.g., [13, 15, 16]). The natural definition [13] by which the solution is a pair of functions $[x(\cdot), q(\cdot)]$ satisfying (2.1) and (2.2) serves the simplest case when $\varphi[a, \varphi(a, p)] = \varphi(a, p)$ (i.e., $a \in H_{\varphi(a, p)} \forall a, p$). In general, the map $p \mapsto \varphi_a(p) := \varphi(a, p)$ can exhibit a whole chain of possible instantaneous transitions of the discrete state for given $a = x(t)$ and $p = q(t)$

$$p =: p_1 \xrightarrow{\varphi_a} p_2 \xrightarrow{\varphi_a} p_3 \xrightarrow{\varphi_a} \dots \xrightarrow{\varphi_a} p_s. \quad (2.4)$$

(The chain is interrupted at the largest index s such that $p_i \neq p_j \forall i \neq j, i, j \leq s$.) Associated with this case are alternative definitions [15, 16]. They, in particular, take into account that, if there exist *cyclic points*, i.e., points $a \in \mathbb{R}^n$ for which, in (2.4), $s \geq 2$ and $\varphi_a(p_s) = p_1$, sliding-mode effects may occur [16]. Further, we shall study the system (2.1), (2.2) in a domain $K \subset \mathbb{R}^n$ and restrict ourselves to the case when

A.3) there are no cyclic points in K .

This, in particular, excludes sliding-mode phenomena and ensures that, in (2.4), $\varphi_a(p_s) = p_s \forall a \in K, p \in Q$. Furthermore, we assume that the system must not perform all the chain (2.4) of instantaneous transitions but may leave it after any transition. So (2.2) is to be replaced by

$$q(t+0) \in \Phi[x(t), q(t)] \quad \text{where} \quad \Phi(a, p) := \left\{ p' \in Q : p' = \varphi_a^{(k)}(p) \text{ for some } k = 1, 2, \dots \right\} \quad (2.5)$$

and $\varphi_a^{(k)} := \underbrace{\varphi_a \circ \dots \circ \varphi_a}_{k \text{ times}}$ is the k -th iteration of the map φ_a . As a result, we arrive at the following

Definition 1. A pair of functions $[x(\cdot), q(\cdot)], x(\cdot) : \Delta \rightarrow \mathbb{R}^n, q(\cdot) : \Delta \rightarrow Q$ (where Δ is an interval) is called the trajectory of the system (2.1), (2.2) if the function $x(\cdot)$ is absolutely continuous, the function $q(\cdot)$ is piece-wise constant and left-continuous, equation (2.1) is true for almost all $t \in \Delta$, and (2.5) is valid for all $t \in \Delta, t \neq \sup\{\theta : \theta \in \Delta\}$.

A.4) The set K is bounded, closed, connected, and invariant, i.e., any trajectory $[x(\cdot), q(\cdot)]$ with $x(0) \in K$ remains in K for $t \geq 0$.

The last assumption will be well-posedness, which means that a small perturbation of the initial data causes only a small perturbation of the trajectory on any bounded time interval. We introduce two definitions of well-posedness. The first one focuses on the continuous state and formally permits the discrete state to be perturbed arbitrarily. The second one forbids this.

Definition 2. The system (2.1), (2.2) is said to be x -well posed (well posed) on K if, for each trajectory $[x(\cdot), q(\cdot)]$, $0 \leq t \leq \tau$ with $x(0) \in K$ and any $\varepsilon > 0$, there exists $\delta > 0$ such that any trajectory $[y(\cdot), p(\cdot)]$ starting in $p(0) = q(0)$ and $y(0) \in K$ with $|y(0) - x(0)| < \delta$ can be defined on $[0, \tau]$ and remains in the ε -neighborhood of the original continuous state $|y(t) - x(t)| < \varepsilon \forall t \in [0, \tau]$ (as well as $\text{mes} \{t : p(t) \neq q(t)\} < \varepsilon$ in the case of well posedness).

Here and throughout, the symbol $\text{mes} E$ stands for the Lebesgue measure of the set E .

Lemma 3. Suppose that Assumptions A.1) – A.4) hold and, for any $p, r \in Q$, $p \neq r$, the differential equations $\dot{x} = f(x, p)$, $\dot{x} = f(x, r)$ have no common integral curves intersecting K . Then the system (2.1), (2.2) is x -well posed on K if and only if it is well posed on K .

A.5) The system (2.1), (2.2) is well posed on K .

This ensures that this system is *deterministic* on K , i.e., any initial data $x(0) \in K$, $q(0) \in Q$ gives rise to a unique trajectory. We close the section with a criterion for Assumption A.5) to be fulfilled. Recall that the symbol $\partial_K E$ denotes the relative boundary of a set $E \subset K$ in K .

Theorem 4. Suppose that Assumptions A.1) – A.4) hold. Then the following statements A) and B) are equivalent.

A) The system (2.1), (2.2) is well posed on K .

B) For any $p, q \in Q$, $p \neq q$, and $a \in \partial_K [T_{p \rightarrow q} \cap K]$,

B.1) the vector $f(a, p)$ looks at $T_{p \rightarrow q}$ at the point a provided $p \in \varphi_a(Q)$; otherwise,

B.2) for any $\varepsilon > 0$, there exists $\delta > 0$ such that the solution $z(\cdot) = z_p(\cdot | a')$ of the Cauchy problem $\dot{z} = f(z, p)$, $z(0) = a'$ starting in $a' \in K \cap H_p$ with $|a' - a| < \delta$ reaches $T_{p \rightarrow q}$ no later than at the time instant $t = \varepsilon$.

Remark. B.1) \Rightarrow B.2). If $z_p(t|a) \in K \forall t \in [0, \eta]$ for some $\eta > 0$, then B.1) \Leftrightarrow B.2).

The property A.5) is fairly co-related with A.3) so far as a cyclic point may cause a chaos on a bounded time interval. To elucidate this, employ the three buffer switched arrival system example from [10]. In other words, consider DA (2.1), (2.2) with $x = (x_1, x_2, x_3)$, $Q = \{1, 2, 3\}$, $f(x, i) := f_i$ where

$f_1 := (2/3, -1/3, -1/3)$, $f_2 := (-1/3, 2/3, -1/3)$, $f_3 := (-1/3, -1/3, 2/3)$, and $\varphi(x, i) := \underline{i-1}$ if $x_{\underline{i-1}} \leq 0$, $\varphi(x, i) := \underline{i-2}$ if $x_{\underline{i-1}} > 0$ and $x_{\underline{i-2}} \leq 0$, otherwise, $\varphi(x, i) := i$. Here $\underline{j} := j$ for $j \geq 1$, $\underline{0} := 3$, and $\underline{-1} := 2$. In correspondence with [10], let us focus attention on the planar invariant domain $K := \{x : x_i \geq 0, x_1 + x_2 + x_3 = 1\}$ (see fig.1).

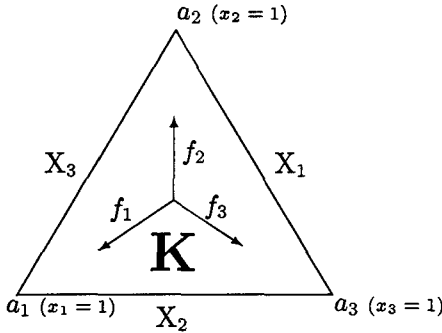


Fig.1

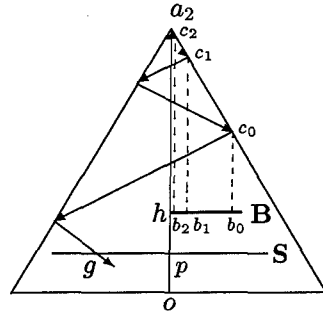


Fig.2

Within the discrete state i , the vector $x \in K$ evolves with the velocity f_i . As soon as it touches the edge $X_j (j \neq i)$ of the triangle K , the state i switches to j . This rule is deterministic except for the vertices of K where a cyclic change of discrete states is offered (e.g., $3 \rightarrow 2 \rightarrow 1 \rightarrow 2$ at the vertex a_3). In [10], a trajectory was assumed to terminate whenever it arrives at a vertex. Fig.2 depicts an infinite family of trajectories starting in $q(0) = 2, x(0) = b_k, k = 0, 1, \dots$. The part of the trajectory until the fall on the edge X_1 is depicted with a dotted line. Then every trajectory runs along a part of a common path depicted with a broken line. Choose a point p on the perpendicular a_2, o . Let $k \rightarrow \infty$. Then the points b_k and c_k approach h and the vertex a_2 , respectively, and $x(T)$ (where $T := (|a_2 - h| + 2|a_2 - p|)/|f_2|$) converges to the point g of the intersection of the above path with the perpendicular S to a_2, o . Obviously, any point $g' \in S$ can be supplied with a sequence of initial states $\{b'_k\} \subset B$ such that $b'_k \rightarrow h$ and $x(T) \rightarrow g'$ as $k \rightarrow \infty$. Then the initial states b_k and b'_k are arbitrarily close to h and to each other provided k is large enough while the corresponding states at $t = T$ are not. This means that, on the bounded time interval $[0, T]$, the behavior of the system is chaotic in the sense that the trajectory is infinitely sensitive to the perturbation of the initial data in the vicinity of h . In view of this, it does not come as a surprise that the behavior of the system on the infinite time interval is also chaotic as shown in [10]. Certainly, not any cyclic point gives rise to a chaos. A criterion to distinguish between those exhibiting and not exhibiting a chaos may be a topic for a separate research. In this paper, we omit this and do not deal, thereby, with cyclic points at all.

3 Decomposition of hybrid dynamical systems

In this section, the interest is focused on decomposition to **DA** for which the only possible behavior of the discrete state is to repeat a fixed chain of transitions $p_1 \mapsto p_2 \mapsto \dots \mapsto p_s \mapsto p_1, p_i \neq p_j \forall i \neq j, i, j \leq s$. It is convenient to identify

such chain with any its cyclic shift in the index. After this, it can be given by a pair $[C, \eta(\cdot)]$ where $C = \{p_1, \dots, p_s\}$ and the map $\eta(\cdot) : C \rightarrow C$ indicates what state follows any $p \in C$, i.e., $\eta(p_i) = p_{i+1}, i = 1, \dots, s-1, \eta(p_s) = p_1$. Taking into account an obvious property of the map $\eta(\cdot)$ results in the following definition.

Definition 5. A pair $[C, \eta(\cdot)]$ is called a cycle in Q if $C \subset Q, C \neq \emptyset, \eta(\cdot) : C \rightarrow C$, and, for each $p \in C^{**}$, the sequence $p, \eta(p), \dots, \eta^{(k-1)}(p)$ ranges over all elements in C and $\eta^{(k)}(p) = p$. Here k is the number of elements in C and $\eta^{(j)}(\cdot)$ is the j -th iteration of the map $\eta(\cdot)$.

Definition 6. A differential automaton is said to be d -autonomous if either $\varphi(x, p) = p$ or $\varphi(x, p) = \eta(p) \forall x \in \mathbb{R}^n, p \in Q$ where $\eta(\cdot) : Q \rightarrow Q$ is a map.

Definition 7. **DA** is said to be cyclic if it is d -autonomous and $[Q, \eta(\cdot)]$ is a cycle.

For a d -autonomous **DA** and any its trajectory, the discrete state transitions

$$p_0 \mapsto p_1 \mapsto p_2 \mapsto \dots \quad (p_j \neq p_{j+1}) \quad (3.6)$$

are independent of the continuous state^{***} and form an $\eta(\cdot)$ -orbit, i.e., $p_{i+1} = \eta(p_i)$. If the system is cyclic, the discrete state q first runs over Q in accordance with the map $\eta(\cdot)$, then returns to p_0 , and, further, merely repeats the above chain of transitions. Underscore, that whole pieces $p_k \mapsto \dots \mapsto p_m$ of the chain (3.6) may be run through instantaneously and this chain may be finite in general. Let us be given several **DA**

$$\dot{x}_i(t) = f_i[x_i(t), q_i(t)], \quad q_i(t+0) = \varphi_i[x_i(t), q_i(t)] \quad (3.7)$$

($i = 1, \dots, l$) with the common continuous state space \mathbb{R}^n and diverse discrete state ones $q_i(t) \in Q_i$.

Definition 8. The system (2.1), (2.2) is called the union of **DA** (3.7) on the domain $K \subset \mathbb{R}^n$ if $Q = Q_1 \cup \dots \cup Q_l$, the sets Q_i do not overlap $Q_i \cap Q_j = \emptyset, i \neq j$, the domain K is invariant for both **DA** (2.1), (2.2) and any system (3.7), and, in (2.1), (2.2),

$$f(x, p) = f_i(x, p), \quad \varphi(x, p) = \varphi_i(x, p) \quad \forall x \in K, p \in Q_i, i = 1, \dots, l. \quad (3.8)$$

If the system (2.1), (2.2) is the union of **DA** (3.7), the set J of all the trajectories $[x(\cdot), q(\cdot)]$ of the system (2.1), (2.2) starting with $x(0) \in K$ is splitted into l pairwise disjoint groups $J = J_1 \cup \dots \cup J_l$ where J_i is the analogous set for **DA** (3.7).

** It suffices to verify this property only for some $p \in C$.

*** which affects only the time instants of transitions

Definition 9. Let only two **DA** (3.7) be given (i.e., $i = 1, 2$ in (3.7)). The second of them (corresponding to $i = 2$) is said to convert on K into the first one in course of time if $Q_1 \subset Q_2$, any trajectory of the first **DA** is a trajectory of the second **DA**, and, conversely, there exists an instant $T > 0$ such that any trajectory of the second **DA** starting with $x(0) \in K$ is a trajectory of the first one in the time domain $t \geq T$.

This means that, if one considers all trajectories $[x(\cdot), q(\cdot)], t \geq 0$ starting with $x(0) \in K$ and restricts any of them on the time interval $[T, +\infty)$, the resultant set of restricted trajectories is common for the both **DA** whereas the first of them is a "subautomaton" of the second one.

The following theorem is the main result of the section.

Theorem 10. Suppose that Assumptions **A.1) – A.5)** hold. On the domain K , **DA** (2.1), (2.2) can be represented as a union of a finite number of d -autonomous **DA** each converting on K into a cyclic **DA** (3.7) (where $i = 1, \dots, l$) in course of time. The number l of these cyclic **DA** as well as the cycle $c_i = [C_i, \eta_i(\cdot)]$ related to any of them are determined uniquely (up to re-arranging in the index i). All the above auxiliary **DA** can be chosen so that they satisfy Assumptions **A.1) – A.5)**.

Definition 11. Any of the above cycles c_i is said to be fundamental for the invariant connected domain K .

It follows from this theorem that, for any trajectory $\vartheta = [x(\cdot), q(\cdot)], x(0) \in K$ of the original system (2.1), (2.2), the discrete state q evolves periodically since some time instant. To explain this in more details, consider such trajectory. The full record (3.6) of all the transitions experienced by the discrete state is called the *discrete path* of the trajectory[†]. To proceed, we need several properties of trajectories. They are revealed by the following lemma.

Lemma 12. Suppose that Assumptions **A.1) – A.5)** hold. For any $a \in K$ and $p \in Q$, the trajectory of the system (2.1), (2.2) starting in $x(0) = a, q(0) = p$ can be defined on $[0, +\infty)$ and is unique. Furthermore, its discrete path is infinite, i.e., the discrete state makes infinitely many transitions on the infinite time interval $[0, +\infty)$. At the same time, it makes only a finite number of transitions on any finite time interval.

Revert now to the foregoing trajectory ϑ . By Theorem 10, there exists an instant $T \geq 0$ such that ϑ is a trajectory of some cyclic **DA** (3.7) in the time domain $t \geq T$. Choose any transition $p_k \mapsto p_{k+1}$ from (3.6) that occurs at an instant $t \geq T$. Then the chain $p_k \mapsto p_{k+1} \mapsto \dots$ evidently obeys the cycle $c_i = [C_i, \eta_i(\cdot)]$ related to the above cyclic **DA** (3.7), i.e., $p_j \in C_i, p_{j+1} = \eta(p_j) \forall j \geq k$. This yields that the discrete path (3.6) does become periodic.

[†] If the states $a := x(t)$ and $p := q(t)$ give rise to a whole chain (2.4) of instantaneous transitions at a moment t , all the transitions from (2.4) that occur in fact must be included in (3.6).

By Theorem 10, analysis of the dynamical behavior of **DA** (2.1), (2.2) in the invariant domain K can be reduced to that concerning several cyclic **DA**. In its turn, any of them can be studied via reduction to discrete-time systems $y_{i+1} = g(y_i)$, $i = 1, 2, \dots$. Indeed, let $[Q, \eta(\cdot)]$ be the cycle associated with this cyclic **DA**. Choose and fix a discrete state $p_0 \in Q$ and, for any $y \in K$, put $g(y) := x(t_1|y, p_0)$ where $\mathfrak{d}_{b,r} = [x(\cdot|b, r), q(\cdot|b, r)]$ is the trajectory starting in $x(0|b, r) = b$, $q(0|b, r) = r$ and t_1 is the least instant $t > 0$ such that $p_0 \neq q(t|y, p_0) \neq q(t+0|y, p_0)$ and the chain (2.4) generated by $p := q(t|y, p_0)$ and $a := x(t|y, p_0)$ contains p_0 . The set $E := \{t \geq 0 : p_0 \neq q(t|y, p_0) \neq q(t+0|y, p_0)\}$ and the chain (2.4) generated by $p := q(t|y, p_0)$ and $a := x(t|y, p_0)$ contains p_0 is countable and has no accumulation points due to Lemma 12, **A.3**), **A.4**), and the definition of cyclic **DA**. So $E = \{t_j\}_{j=1}^\infty$ where $t_j < t_{j+1}$ and $t_j \rightarrow \infty$ as $j \rightarrow \infty$. Since the system is deterministic on K , we have $y_{j+1} = g(y_j)$ where $y_j := x(t_j)$, $j = 1, 2, \dots$. So the behavior of trajectories \mathfrak{d}_{a,p_0} as $t \rightarrow \infty$ is fairly co-related with that of trajectories $\{y_j\}$ of the system $y_{j+1} = g(y_j)$ as $j \rightarrow \infty$.

Analysis of this system necessarily employs certain properties of the function $g(\cdot)$. The following theorem yields easily that, under the circumstances, this function is at least continuous. It also demonstrates that well-posedness in the sense of Definition 2 implies well-posedness in a far stronger sense.

Theorem 13. *Suppose that Assumptions **A.1**) – **A.5**) are valid and $p \in Q$ is given.*

The discrete path $\{p_j\}_{j=0}^\infty$ of the trajectory $\mathfrak{d}_{a,p} = [x(\cdot|a, p), q(\cdot|a, p)]$ is independent of $a \in K$. Given $a \in K$ and $j = 1, 2, \dots$, denote by $\tau_j(a)$ the time instant when the system starting in $x(0) = a$, $q(0) = p$ makes the transition $p_{j-1} \mapsto p_j$. Then the functions $\tau_j(\cdot) : K \rightarrow [0, +\infty)$, $j = 1, 2, \dots$ are continuous and

$$q(t|a, p) = p_j \quad \forall t \in (\tau_j(a), \tau_{j+1}(a)], a \in K, j = 0, 1, \dots \quad (\tau_0(a) := 0), \quad (3.9)$$

$$\max_{t \in [0, \lambda]} |x(t|a', p) - x(t|a, p)| \rightarrow 0 \quad \text{as } a' \rightarrow a, a' \in K \quad \text{for all } \lambda > 0, a \in K. \quad (3.10)$$

4 Existence of periodic trajectories

As an example of employing the above reduction, we establish existence of periodic trajectories for the system (2.1), (2.2). On contrary to Section 3, now we are interested in trajectories that are periodic in not only the discrete but also the continuous state. It will be shown that such trajectories exist. Moreover, among them, there necessarily are those with relatively simple structure. To this end, we start with an insight on the structure of periodic trajectories.

Assume that Assumptions **A.1**) – **A.5**) are fulfilled. Let $\mathfrak{d} = [x(\cdot), q(\cdot)]$ be a periodic trajectory with $x(0) \in K$ and T be its least period, i.e., $T := \inf \{\tilde{T} : \tilde{T} > 0, x(t+\tilde{T}) = x(t), q(t+\tilde{T}) = q(t) \forall t \geq 0\}$. (Here $T > 0$, because, otherwise, $q(\cdot) \equiv p = \text{const}$ and the periodic function $x(\cdot)$ would satisfy the equation

$\dot{x} = f(x, p)$ in violation of **A.1**.) The discrete path $p_0 \mapsto p_1 \mapsto \dots \mapsto p_s$ of this trajectory on the interval $0 < t \leq T$ is finite by Lemma 12. Due to Theorem 10, it obeys some fundamental cycle $[C, \eta(\cdot)]$ of the domain K , i.e., $p_i \in C \forall i, p_{i+1} = \eta(p_i), i = 0, \dots, s-1$. Periodicity and **A.1**) imply that the above path is composed of $k \geq 1$ complete runs through that cycle. The periodic trajectory is said to be *elementary* (for the invariant domain K) if $k = 1$ for it. For such trajectory, the least period of the discrete state alone evidently equals the above period T . So is the least period of the continuous state if, for any $p, r \in Q, p \neq r$, the equations $\dot{x} = f(x, p), \dot{x} = f(x, r)$ have no common integral curves intersecting K .

Theorem 14. *Suppose that Assumptions **A.1**) – **A.5**) hold and the domain K is homeomorphic (see [22, p.188] for the definition) to a closed ball. Then the domain K contains an elementary (for K) periodic trajectory $\mathfrak{d} = [x(\cdot), q(\cdot)], x(t) \in K \forall t \geq 0$. Moreover, for any fundamental cycle $c = [C, \eta(\cdot)]$ of the domain K , there exists an elementary periodic trajectory \mathfrak{d} that lies in K , i.e., $x(t) \in K \forall t \geq 0$, and obeys this cycle, i.e., $p_j \in C, p_{j+1} = \eta(p_j), j = 0, 1, \dots$ where $\{p_j\}$ is the discrete path of \mathfrak{d} .*

5 An analog of the Poincaré-Bendixon theorem

In this section, we consider the planar system (2.1), (2.2), i.e., $x(t) \in \mathbb{R}^2$ in (2.1), (2.2). Suppose that the system satisfies Assumption **A.2**) in the following strengthened form **A.6**) and the following additional assumption **A.7**) is also valid. Recall that the symbol $\partial_K E$ stands for the relative boundary of a set $E \subset K$ in K .

- A.6)** *For $p, r \in Q, p \neq r$, and $a \in \partial_K(T_{p \rightarrow r} \cap K)$, one of the vectors $f(a, p), -f(a, p)$ looks at $H_p := \{x : \varphi(x, p) = p\}$ with the other looking at $T_{p \rightarrow r}$ at the point a .*
- A.7)** *The domain K is closed. Given $p, r \in Q, p \neq r$, the set $T_{p \rightarrow r} := \{x : \varphi(x, p) = r\}$ is closed. Its relative boundary $\partial_K(T_{p \rightarrow r} \cap K)$ can be partitioned into a finite set $\mathfrak{S}_{p \rightarrow r}$ of compact segments of C^1 -smooth non-selfcrossing curves so that no more than two segments have a common point and any two segments either do not intersect or have a common end-point and are transversal. Any two segments corresponding to different pairs (p, r) and (p', r') are either disjoint or transversal at any common point or lie on a common C^1 -smooth curve.*

The main result of the section is a necessary and sufficient criterion for the system (2.1), (2.2) to exhibit a simple dynamics on the invariant domain K . Roughly speaking, the simple dynamics is a nonchaotic dynamics like that described in the classic Poincaré-Bendixon theorem [22, p.295]. However, unlike this theorem, stationary points are impossible under the circumstances. (They are impossible within a given discrete state because of **A.1**) and within a sliding mode since it cannot occur due to **A.3**), see [16] for details.) In the absence of

stationary points, the Poincaré-Bendixon theorem states that any trajectory of a stationary ordinary differential equation either 1) is periodic or 2) converges to one of no more than countably many limit cycles. (Recall, that, in general, periodic trajectories may fill certain domains.) For the system (2.1), (2.2) the phenomenon 1) can appear in more general fashion. A trajectory can be nonperiodic but become periodic since some time instant (so far as the system is not deterministic with respect to the backward direction of time in general). Furthermore, the simplicity of the dynamics is supposed to mean that, for periodic trajectories, the variety of possible discrete state behaviors is finite. This concerns trajectories that are periodic both precisely and approximately. To precise details, introduce the following definition.

Definition 15. Let $\vartheta = [x(\cdot), q(\cdot)]$ be a trajectory defined on an interval Δ . The sequence $\{p_i\}$ of values taken by the discrete state $q = q(t)$ while t runs increasingly through Δ is called the symbolic range of the trajectory ϑ .

Unlike the notion of the discrete path, the discrete states through which $q(\cdot)$ runs instantaneously are not taken into account now.

The symbolic range of a periodic trajectory ϑ on $(0, +\infty)$ results from periodic repetition of its symbolic range $\sigma_\vartheta = (p_0, \dots, p_k)$ on $(0, T]$ where $T > 0$ is the least positive period. The property in question means that the variety of symbolic ranges σ_ϑ of periodic trajectories is finite.

In the Poincaré-Bendixon theorem, the limit cycle is treated geometrically as a curve in \mathbb{R}^2 . Correspondingly, the convergence to it means that the distance to the curve converges to the zero. Because of the discrete state, it is convenient now not to employ geometrical treatment but to define the convergence to a limit cycle in terms of the periodic trajectory related to it. In doing so, it must be taken into account that the converging and, respectively, limit trajectories must not become synchronous as $t \rightarrow \infty$. Let $\vartheta = [x(\cdot), q(\cdot)]$ be a periodic trajectory with the period $T > 0$. We say that a trajectory $[y(\cdot), p(\cdot)], t \in [0, \infty)$ converges to ϑ as $t \rightarrow \infty$ if there exists a sequence $\{\tau_i\} \subset (0, +\infty)$ such that $\tau_{i+1} - \tau_i \rightarrow T$ as $i \rightarrow \infty$ and

$$\begin{aligned} \max_{t \in [0, \lambda]} |y(t + \tau_i) - x(t)| &\rightarrow 0, \\ \text{mes} \{t \in [0, \lambda] : p(t + \tau_i) \neq q(t)\} &\rightarrow 0 \end{aligned} \quad \text{as } i \rightarrow \infty \quad \forall \lambda > 0. \quad (5.11)$$

Here and throughout, $\text{mes } E$ stands for the Lebesgue measure of a set E . Note also that, under the circumstances, the trajectory starting with $x(0) \in K$ can be defined on $[0, +\infty)$ and is unique.

Definition 16. The system (2.1), (2.2) is said to exhibit a simple periodic dynamics on K if

- i) there exists no more than a countable set \mathfrak{P} of periodic trajectories lying in K such that any trajectory ϑ starting in K either is periodic for $t \geq T_\vartheta$ (where the time T_ϑ may depend on ϑ in general) or converges to some trajectory from \mathfrak{P} and never becomes periodic itself;

- ii) there exists a time $T_* > 0$ such that, for any trajectory lying in K , its symbolic range $\{p_j\}$ on the time interval $[T_*, \infty)$ is periodic (in the index j) and all such trajectories in total give rise to a finite number of ranges $\{p_j\}$.

Let $c = [C, \eta(\cdot)]$ be a fundamental cycle of K . The set

$$S_c := \{(a, p) : p \in C, a \in \partial_K (T_{p \rightarrow \eta(p)} \cap K)\}. \quad (5.12)$$

is called the *c-skeleton* of the domain K .

Restructuring points. Given $p \in C$, put

$$S_p^{(1)} := \partial_K (T_{p \rightarrow \eta(p)} \cap K), \quad S_p^{(i+1)} := S_p^{(i)} \cap T_{\eta^i(p) \rightarrow \eta^{i+1}(p)} \quad (5.13)$$

where $i = 1, 2, \dots$ and $\eta^i(\cdot)$ is the i -th iteration $\eta(\cdot) \circ \dots \circ \eta(\cdot)$ of the map $\eta(\cdot)$. In terms of the chain

$$p_0 := p \mapsto p_1 := \varphi_a(p_0) \mapsto \dots \mapsto p_n := \varphi_a(p_{n-1})^\dagger \quad (5.14)$$

of instantaneous transitions generated by a and p , (5.13) shapes into

$$S_p^{(i)} := \left\{ a \in S_p^{(1)} : \begin{array}{l} \text{the chain (5.14) begins with} \\ \text{the sequence } p \mapsto \eta(p) \mapsto \dots \mapsto \eta^i(p) \end{array} \right\}.$$

By A.4), $S_p^{(i)} = \emptyset \forall i \geq k$ where k is the number of elements in C .

Lemma 17. Any nonempty set $S_p^{(i)} (p \in C, i = 1, 2, \dots)$ is a union of a finite number of points and pair-wise disjoint topological segments (i.e., sets homeomorphic[§] to an interval $[t_0, t_1], t_0 < t_1$).

As a result, the relative boundary ∂_p^i of $S_p^{(i+1)}$ in $S_p^{(i)}$ is a finite set. Any point $(a, p) \in S_c$ such that $a \in \partial_p^i$ for some $i = 1, 2, \dots$ is called the *restructuring point* on the *c-skeleton*.

Regular and singular points. For $(a, p) \in S_c$, (5.12) $\Rightarrow a \in \partial_K (T_{p \rightarrow \eta(p)} \cap K)$ and, by A.7), a belongs to one s or two s', s'' segments from $\mathfrak{S}_{p \rightarrow \eta(p)}$. In the first case, the line tangent to s at a is said to be *tangent to $\partial_K (T_{p \rightarrow \eta(p)} \cap K)$ at a* . In the second case, the angle formed by the rays tangent to s' and s'' is said to be *tangent to $\partial_K (T_{p \rightarrow \eta(p)} \cap K)$ at a* . Consider the last term p_n in the chain (5.14). The pair $(a, p) \in S_c$ is said to be *regular* if the line spanned by a and $a + f(a, p_n)$ intersects the both open domains into which the line or angle tangent to $\partial_K (T_{p \rightarrow \eta(p)} \cap K)$ at a splits the plane \mathbb{R}^2 . A pair $(a, p) \in S_c$, which is not regular, is said to be *singular*.

Backstepping mapping in the *c-skeleton*. Let $z_r(\cdot|a)$ (where $r \in Q$ and $a \in \mathbb{R}^2$) stand for the solution $z(\cdot)$ of the boundary problem $\dot{z} = f(z, r), z(0) = a$ extended on the maximal interval. Given $\omega = (b, r) \in S_c$, put

$$E_\omega := \{\theta < 0 : z_r(t|b) \in H_r \quad \forall t \in [\theta, 0]\}, \quad (5.15)$$

[†] The chain is interrupted at the largest index n such that $p_i \neq p_j \forall i \neq j, i, j \leq n$.

[§] See [22, p.188] for the definition

$$W(\omega) := \{(a, p) \in S_c : (a, p) = \omega \text{ or } p_n = r \text{ in (5.14) and } a = z_r(\theta|b) \text{ for some } \theta \in E_\omega\}. \quad (5.16)$$

The multivalued function $(b, r) \in S_c \mapsto W(b, r) \subset S_c$ is called the *backstepping mapping in the c-skeleton*. The following lemma illustrates this notion.

Lemma 18. *Let $(a, p) \in S_c, (b, r) \in S_c, (a, p) \neq (b, r)$. Then $(a, p) \in W(b, r)$ if and only if there exists a trajectory $[x(\cdot), q(\cdot)], t \geq 0$ lying in $K \times C$ and two consequent discontinuity points $0 \leq t_1 < t_2$ of the function $q(\cdot)$ such that $(a, p) = [x(t_1), q(t_1)]$ and $(b, r) = [x(t_2), q(t_2)]$.*

For $\omega \in S_c$, denote $W^1(\omega) := W(\omega), W^{i+1}(\omega) := \{\omega' : \omega' \in W(\omega_*) \text{ for some } \omega_* \in W^i(\omega)\}, i = 1, 2, \dots$. The set $O^-(\omega) := \bigcup_{i=1}^\infty W^i(\omega)$ is called the *backward c-orbit* of the point ω .

Our last assumption is as follows.

A.8) *For each fundamental cycle c of the invariant domain K , the set of the singular points on the c-skeleton is finite and the backward c-orbit of any such point is also finite.*

The following theorem is the main result of the section.

Theorem 19. *Let $x(t) \in \mathbb{R}^2$ in (2.1), (2.2) and Assumptions A.1)–A.8) hold. Then the following two statements are equivalent.*

- (i) *The system (2.1), (2.2) exhibits a simple periodic dynamics on K .*
- (ii) *For each fundamental cycle c of the invariant domain K , the backward c-orbit of any restructuring point on the c-skeleton is finite.*

Assumption **A.8)** is evidently fulfilled if there are no singular points on the c-skeleton for any c . This is the case if, for example, each set $\mathfrak{S}_{p \rightarrow r}$ either consists of pair-wise disjoint straight segments or is empty and any vector-field $f(\cdot, p), p \in Q$ is constant and transversal to all of the above segments.

In the remainder of the section, (i) of Theorem 19 and Assumptions **A.1)–A.8)** are assumed to be true.

Remark 1. *Let, for each fundamental cycle $c = [C, \eta(\cdot)]$ of K , any segment $s \in \mathfrak{S}_{p \rightarrow \eta(p)}$ lie on an analytical curve and the vector fields $f(\cdot, p), p \in C$ be analytical. Then, in i) of Definition 16, the set \mathfrak{P} can be chosen finite.*

Remark 2. *In i) of Definition 16, the time T_d can be chosen independent of d .*

Let d be a trajectory lying in K . Recall that the full record $p_0 \mapsto p_1 \mapsto \dots$ of all the transitions experienced by the discrete state is called the *discrete path* of d . Under the circumstances, his path is evidently composed of successive gearing chains $\mathcal{P}_1, \mathcal{P}_2, \dots$

$$\underbrace{p_0 \mapsto \dots \mapsto p_{k_1}}_{\mathcal{P}_1} \mapsto \dots \mapsto \underbrace{p_{k_2} \mapsto \dots \mapsto p_{k_3}}_{\mathcal{P}_3} \mapsto \dots \quad (5.17)$$

(where $1 \leq k_1 < k_2 < \dots$) each encompassing all the transitions that occur at a certain time instant. It easily follows from Theorem 10 that the path $\{p_j\}$ is eventually periodic. In view of A.3), Theorem 19 ensures via justifying ii) of Definition 16 that the distribution (5.17) also becomes eventually (namely, for $t > T_*$) periodic, i.e., $\mathcal{P}_{j+k} = \mathcal{P}_j \forall j \approx \infty$.

Acknowledgement This work was supported by the Australian Research Council.

References

1. "Hybrid systems I. Verification and control", ed. Grossman, R.L., Nerode, A., Ravn, A.P., and Rishel, H. Lecture Notes in Computer Sciences, vol.736, Springer, 1993.
2. "Hybrid systems II. Verification and control", ed. Antsaklis, P., Kohn, W., Nerode, A., and Sastry, Sh. Lecture Notes in Computer Sciences, vol.999, Springer, 1995.
3. "Hybrid systems III. Verification and control", ed. Alur, R., Henzinger, T.A., and Zontag, E.D. Lecture Notes in Computer Sciences, vol.1066, Springer, 1996.
4. Antsaklis, P., Stiver, J., and Lemmon, M., "Hybrid systems modelling and autonomous control systems", in [1], pp.366-392.
5. Brockett, R.W., "Hybrid models for motor control systems". In Trentelman, H.L and Willems, J.C., editors, "Essays in Control", Birkhauser, Boston, 1993.
6. Gollu, A. and Varaiya, P., "Hybrid dynamical systems", In Proc. of the 28th IEEE Conf. On Decision and Control. Tampa. 1989.
7. Savkin, A. V., Petersen, I. R., Skafidas, E., and Evans, R.J., "Hybrid dynamical systems: robust control synthesis problems", Systems & Control Letters, vol.29, pp.81-90, 1996.
8. Savkin, A. V., Evans, R.J., and Petersen, I. R., "A new approach to robust control of hybrid dynamical systems", in [3], pp.553-562.
9. Varaiya, P., "Smart cars on smart roads. Problems of control", IEEE Trans. on Autom. Control, vol.38, no.2, pp.195-207, 1993.
10. Chase, C., Serrano, J., and Ramadge, P., "Periodicity and chaos from switched flow systems: contrasting examples of discretely controlled continuous systems", IEEE Trans. on Autom. Control, vol.38, no.1, pp.70-83, 1993.
11. Alur, R., Couroubetis, C., Henzinger, T., and Ho, P., "Hybrid automata: An algorithmic approach to the specifying and verification of hybrid system", in [1], pp.209-229.
12. Gennaro, S., Horn, C., Kulkarni, S., and Ramadge, P., "Reduction of timed hybrid systems". Proc. IEEE Conf. on Decision and Control, pp.4215-4220, FL, 1994.
13. Tavernini, L., "Differential Automata and their Discrete Simulators", *Nonlinear Analysis. Theory. Methods & Applications*, vol. 11, no.6, pp.665-683, 1987.
14. Filippov, A.F., "Differential Equations with Discontinuous Righthand Sides", Kluwer Academic Publishers, 1988.
15. Doğruel, M. and Özgüner, U., "Modelling and stability issues in hybrid systems" in [2], pp.148-165.
16. Drakunov, S., "Sliding modes in hybrid systems - a semigroup approach", Proc. of the 33rd Conference on Decision and Control, Florida, 1994, pp.4235-4240.
17. Ushio, T., Ueda, H., and Hirai, K., "Controlling chaos in a switched arrival system", Systems & Control Letters, vol.26, pp.335-339, 1995.

18. Horn, C. and Ramadge, P.J., "Dynamics of switched arrival system with thresholds", Proc. of the 32nd CDC, San Antonio, Texas, 1996, pp.288-293.
19. Horn, C. and Ramadge, P.J., "A topological analysis of a family of dynamical systems with non-standard chaotic and periodic behavior", International Journal of Control, vol.67, No.6, pp.979-1020, 1997.
20. Asarin, E., Maler, O., and Pnueli, A., "Reachability analysis of dynamical systems having piecewise-constant derivatives", Theoretical Computer Sciences, vol.138, pp.35-65, 1995.
21. Guckenheimer, J., "Planar hybrid systems", in [2], pp.202-225.
22. Petrovski, I.G., "Ordinary differential equations". Pover Publications Inc., N.Y., 1966.

Optimization of Generalized Solutions of Nonlinear Hybrid (Discrete-Continuous) Systems*

Boris M. Miller

Institute for Information Transmission Problems,
Russian Academy of Sciences
19 B. Karetny per., Moscow, GSP-4, 101447, Russia,
E-mail:bmiller@ippi.ac.msk.su.

Abstract. The optimal control problem for hybrid (discrete-continuous) system is considered in the case when the continuous behavior can be controlled and discontinuities arise when the system achieves the boundary of some set. We suppose that discontinuities can be considered as a result of some impulsive inputs, which can be represented in feedback form as the intermediated conditions. Meanwhile, various types of irregularities such as: nonextendability of solution or sliding mode can arise. However, if the jumps of solution are described by some shift operator, as for hybrid system satisfying the robustness condition, one can reduce this problem to the standard problem of nonsmooth optimization and the representation of solution by differential equation with a measure and the existence theorem for optimal solution can be obtained.

1 Introduction

In recent years there has been a significant increase in modelling and control of hybrid systems, which are frequently can be treated as systems characterized by continuous and discrete behaviour. The motion of such systems can be divided into regular and singular parts, i.e. continuous and jumping, respectively. These systems are very typical for various mechanical applications, where the discrete-continuous modes of motions could arise because of shocks and friction. There has been a significant progress in this area, including the development of the rigorous mathematical framework for the description of these systems and preliminary formulations of the procedures for synthesis of control laws for them. However, the common mathematical feature of these class of systems is the presence of singularities, which manifest themselves in: discontinuities

* This work was supported in part by National Science Foundation of USA grant CMS 94-1447s and International Association for the Promotion of Cooperation with Scientists from the Independent States of the Former Soviet Union (INTAS) grants 94-697 and 93-2622.

and nonsmoothness in system motion, jumps in system dimension, the lack of the continuous dependence on initial conditions and nonuniqueness of solution of equations of motion (see, for example, [1], [3]).

Traditionally the control of such systems has been exerted either during the nonsingular phase of the system motion or during the singularity phase, which was induced by the control action itself and did not exist in the system naturally [5], [6], [10], [13]. Modern theory of impulsive control provides the appropriate framework for the synthesis of the impulsive control actions in the open loop form. The proper tool for the description and optimization of such systems is the discontinuous time transformation method, developed for nonlinear systems in [5], [6]. However, this approach cannot be directly applied to general hybrid systems, where impulsive actions can arise as feedback ones, when the system under control achieves the appropriate state or the set of states.

This paper focuses on the novel idea of considering a jump as a result of some "fictitious motion" along the paths of some auxiliary system, which provides a model of "fast motion" and describes the jump, arising in the motion of hybrid system, in terms of some shift operator. This approach bases on the representation of robust hybrid system, which was obtained in [8], where hybrid systems are treated as systems with impulsive inputs. However, if we consider these systems as ones with impulsive actions in feedback form, it becomes necessary to find a more general mathematical framework, than for standard problems with impulsive controls.

Thus, the goals of this paper are:

- to develop the mathematical framework for the description of controllable hybrid systems with impulsive actions in feedback form;
- to derive the appropriate equations for the description of motion;
- on the basis of this framework to develop procedures for the design of control in these systems to satisfy specific control objectives.

2 Problem statement

Consider the evolution of discrete-continuous dynamical system, whose behaviour be described on some interval $[0, T]$ by variable $X(t) \in R^n$, which satisfies the differential equation

$$\dot{X}(t) = F(X(t), u(t)), \quad (1)$$

with given initial condition $X(0) = x_0 \in R^n$ and following intermediate conditions

$$X(\tau_i) = X(\tau_i-) + \Psi(X(\tau_i-)), \quad (2)$$

which are given for some sequence of instants $\{\tau_i, i = 0, \dots, N\}$, $N \leq \infty$, satisfying the recurrence conditions

$$\begin{aligned} \tau_0 &= 0 \\ \tau_i &= \begin{cases} \inf\{\tau_{i-1} < t \leq T : G(X(t-)) = 0\}, \\ \infty, & \text{if the appropriate set is empty.} \end{cases} \end{aligned} \quad (3)$$

In equation (2) $X(\tau_i-) = \lim_{t \uparrow \tau_i} X(t)$, and τ_i is the sequence of instants when the system states change discontinuously.

So, the state of system changes continuously in halfintervals $[0, \tau_1)$, ... $[\tau_{i-1}, \tau_i)$, ... $[\tau_N, T]$, and undergoes a sudden change at every instant τ_i , whose value, due to equation (2), depends on the state preceding the jump.

We suppose that control variable in (1)

$$u \in U \subset R^m, \quad (4)$$

where U is some compact set, and function $F(X, u)$ be continuous with respect to all variables and continuously differentiable with respect to X . To be sure that solution of (1) is continuable to the right we need some additional assumptions concerning the functions $\Psi(X)$ and $G(X)$. So we suppose that $X(\tau_i) = X(\tau_i-) + \Psi(X(\tau_i-))$ is the result of the action of shift operator along the paths of differential equation

$$\dot{y}(s) = B(y(s)), \quad s \in [0, \infty) \quad (5)$$

with initial condition $y(0) = X(\tau_i-)$.

Therefore, if $\Phi(x, s)$ is the general solution of (5) with initial condition $y(0) = x$, then

$$X(\tau_i) = X(\tau_i-) + \Psi(X(\tau_i-)) = \Phi(X(\tau_i-), s^*(X(\tau_i-))), \quad (6)$$

where

$$s^*(X(\tau_i)) = \inf\{s > 0 : G(\Phi(X(\tau_i-), s)) = 0\}, \quad (7)$$

and on the interval $(0, s^*(X(\tau_i)))$ we have the relation $G(\Phi(X(\tau_i-), s)) > 0$.

We assume also that $B(y)$ be continuously differentiable with respect to y .

Remark 1. All these conditions are not sufficient to prove the continuability of solution for arbitrary initial condition x_0 , and some measurable control $u(\cdot)$, however, if there exists some bounded solution $X(t)$ with finite number of jumps one can establish its uniqueness and continuous dependence from initial conditions.

The optimization problem to be considered is the minimization of performance criterion

$$J[X(\cdot), u(\cdot)] = \phi_0(X(T)) \quad (8)$$

with some continuously differentiable function ϕ .

Remark 2. In spite of all our assumptions concerning regularity of functions involved into the problem statement, this problem belongs to a class of extremely irregular ones due to the possibility of nonextendability of solution (if the set of points s in (7) is empty or infimum equal to infinity). The first case leads to a so-called "sliding mode" along the set $G(x) = 0$, like in systems with discontinuous right-hand-side [11], the second one corresponds the case of nonextendability of solution beyond the some point of jump. However, in the case when the jump behavior is described by some shift type operator one possible to reduce this problem to the more regular one by using the discontinuous time transformation as in impulsive control problems [5].

Remark 3. The description of jump by some shift operator looks like rather artificial, however, all discrete-continuous systems that are stable (or robust) with respect to an approximation procedure of impulsive input admit such jump representation [6], [10], [13].

Before further consideration it would be useful to present a simple example of hybrid system which is rather typical one and simultaneously has all specific features of the systems described above.

Example. Dynamic of point with elastic shocks.[9] Consider a motion of the unit mass point with generalized coordinates $\{x_1, x_2\}$ (state and velocity, respectively), which moves along the straight line till the elastic obstacle at the point $x_1 = 0$. Suppose that initial state $x_1(0) < 0$, and the force depends on the state, velocity and some control u , that is in the area $x_1 < 0$, the motion equations are

$$\begin{aligned} \dot{x}_1(t) &= x_2(t), \\ \dot{x}_2(t) &= F(x_1(t), x_2(t), u(t)). \end{aligned} \quad (9)$$

The elastic shock at instant $\{\tau : x_1(\tau) = 0\}$ causes the sudden change of velocity sign, i.e.

$$x_2(\tau) = x_2(\tau-),$$

and the instant evolution can be described by the appropriate shift operator along the paths of the system of differential equations

$$\begin{aligned}\dot{y}_1(s) &= y_2(s) \\ \dot{y}_2(s) &= -y_1(s)\end{aligned}\tag{10}$$

with initial conditions

$$y_1(0) = x_1(\tau-), \quad y_2(0) = x_2(\tau-).$$

Indeed, the solution of above system is

$$y_1(s) = x_2(\tau-) \sin(s), \quad y_2(s) = x_2(\tau-) \cos(s),$$

hence, we have $y_1(s) > 0$ on the interval $(0, \pi)$ and

$$y_1(\pi) = 0, \quad y_2(\pi) = -y_2(0) = -x_2(\tau-).$$

Therefore, the shift operator along the paths of system (10) describes the jump behaviour in proper way.

3 Description of solution via discontinuous time transformation

Suppose for some control $u(\cdot)$ we have any solution of (1) defined on the interval $[0, T]$, and suppose also, that this solution, namely $X(t)$, has a finite number of jumps at points $\{\tau_i, i = 1, \dots, N\}$. It means that for every $i = 1, \dots, N$ be defined the set of $s_i^* = s^*(X(\tau_i-)) < \infty$, such that

$$X(\tau_i) = \Phi(X(\tau_i-), s^*(X(\tau_i-))).$$

Consider the time interval $[0, T_1]$, where

$$T_1 = T + \sum_{i=1}^N s_i^*,\tag{11}$$

and define on $[0, T_1]$ function

$$\alpha(s) = \begin{cases} 1, & \text{if } s \in [\tau_i + \sum_{k < i} s_i^*, \tau_i + \sum_{k \leq i} s_i^*) \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

Define on the interval $[0, T_1]$ the auxiliary system

$$\begin{aligned} \dot{y}(s) &= \alpha(s)F(y(s), u(\eta(s))) + (1 - \alpha(s))B(y(s)), \\ \dot{\eta}(s) &= \alpha(s) \end{aligned} \quad (13)$$

with initial conditions

$$y(0) = x_0, \quad \eta(0) = 0.$$

Then the following correspondence exists between auxiliary and original system (1).

Theorem 1. *For any solution $X(t)$ of (1) define function $\eta(s)$ by relation (12) and the inverse one by relation*

$$\Gamma(t) = \inf\{s : \eta(s) > t\},$$

with $\Gamma(T) = T_1$ by definition.

If $\{y(\cdot), \eta(\cdot)\}$ be the solution of (13), then

$$X(t) = y(\Gamma(t))$$

is the solution of system (1).

The proof follows from result, which have been obtained for discontinuous time transformation in robust discrete-continuous systems [6].

Notice that by definition $\alpha(s) = 1$ if $G(y(s)) < 0$ and $\alpha(s) = 0$ if $G(y(s)) > 0$. This observation will be a basis for further transformation of original optimization problem. Moreover, we could consider the system (13) as a system, which in some sense is equivalent to (1), with new variable $\eta(s)$ for the rescaled time and with the variable α as an additional control, which satisfies the constraint

$$\alpha(s) = \begin{cases} 1 & \text{if } G(y(s)) < 0, \\ 0 & \text{if } G(y(s)) > 0. \end{cases} \quad (14)$$

However, (14) does not define α on the boundary of constraint, namely on the set $\{G(y(s)) = 0\}$, but here we can admit the relaxation of the problem, putting $\alpha(s) \in [0, 1]$. This relaxation corresponds to a standard method of convexification of right-hand-side to guarantee the existence of the optimization problem solution [4].

Moreover, one can present the constraint (14) in the integral form, i.e.

$$\begin{aligned} \int_0^{T_1} \alpha(s) G^+(y(s)) ds &= 0, \\ \int_0^{T_1} (1 - \alpha(s)) G^-(y(s)) ds &= 0, \end{aligned} \quad (15)$$

where

$$G^+(y) = \min\{G(y), 0\}, \quad G^-(y) = \min\{-G(y), 0\}.$$

Using this relaxation one can obtain the following result, which generalizes Theorem 1.

Theorem 2. *Let $\{y(\cdot), \eta(\cdot)\}$ be any solution of system (13), with some Lebesgue measurable controls $\{\alpha(\cdot), u_1(\cdot)\}$, satisfying constraints*

$$\alpha(s) \in [0, 1], \quad u_1(s) \in U \quad \text{a. e. on } [0, T_1], \quad (16)$$

and such that $y(\cdot)$ satisfies (15) and $\eta(T_1) = T$. Define monotonically increasing $\Gamma(t)$ by relation

$$\Gamma(t) = \inf\{s : \eta(s) > t\}, \quad \Gamma(T) = T_1, \quad (17)$$

then for $X(t) = y(\Gamma(t))$ there exist:

1. *Lebesgue measurable control $u(\cdot) : u(t) \in U$ almost everywhere on $[0, T]$,*
2. *nonnegative regular measure $\mu(dt)$, localized on the set $\{t : G(X(t)) = 0\}$ and having the Lebesgue decomposition*

$$\mu((0, t]) = \mu^c((0, T]) + \sum_{\tau \leq t} \mu(\{\tau\}),$$

where $\mu^c(dt)$ is continuous component of measure, and $\mu(\{\tau\})$ is a discrete one, localized at point τ ;

such that $X(t)$ is the unique right continuous solution of equation with a measure

$$dX(t) = F(X(t), u(t))dt + B(X(t))d\mu^c(dt) + \sum_{\tau \leq t} \Psi(X(\tau-), \mu(\{\tau\})), \quad (18)$$

satisfying the constraint

$$G(X(t)) \leq 0, \quad \text{for any } t \in [0, T]. \quad (19)$$

Proof. Define $\Gamma(\cdot)$ by relation (17). Then Γ be monotonically increasing and right continuous [6]. Therefore, $X(t) = y(\Gamma(t))$ is right continuous and satisfies the equation

$$\begin{aligned} X(t) = x_0 + \int_0^{\Gamma(t)} \alpha(s) F(y(s), u_1(s)) ds + \\ \int_0^{\Gamma(t)} (1 - \alpha(s)) B(y(s)) ds. \end{aligned} \quad (20)$$

Define the distribution function of $\mu(dt)$ by relation

$$\mu((0, t]) = \int_0^{\Gamma(t)} (1 - \alpha(s)) ds,$$

then

$$\mu^c((0, t]) = \int_0^{\Gamma(t)} I\{s : \eta(s) \in D_\Gamma\} (1 - \alpha(s)) ds,$$

$$\mu(\{\tau\}) = \Gamma(\tau) - \Gamma(\tau-), \quad \text{if } \tau \in D_\Gamma,$$

where D_Γ is the set of jump points of Γ and symbol $I\{A\}$ stands for the indicator function of the set A .

Assuming $u(t) = u_1(\Gamma(t))$, we obtain it to be Lebesgue measurable (see [7] Thm. 4.1), and applying the same arguments of time substitution as in [6] we yield that $X(t)$ satisfies the equation

$$\begin{aligned} X(t) = x_0 + \int_0^t F(X(s), u(s)) ds + \\ \int_0^t B(X(s)) d\mu^c(s) + \sum_{\tau \in D_\Gamma \cap \{\tau \leq t\}} \int_{\Gamma(\tau-)}^{\Gamma(\tau)} B(y(s)) ds, \end{aligned} \quad (21)$$

where

$$\int_{\Gamma(\tau-)}^{\Gamma(\tau)} B(y(s))ds = \Psi(y(\Gamma(\tau-)), \Gamma(\tau) - \Gamma(\tau-)) = \Psi(X(\tau-), \mu(\{\tau\}))$$

due to relations (5) and (6).

Since (21) is the integral representation of (18), we have proved that $X(t)$ satisfies the above equation. Uniqueness of solution follows in standard way from differentiability of F and B .

Now by applying the time substitution to relations (15), we obtain

$$\int_0^{T_1} \alpha(s) G^+(y(s)) ds = \int_0^T G^+(X(s)) ds = 0,$$

$$\int_0^{T_1} (1 - \alpha(s)) G^-(y(s)) ds = \int_0^T G^-(X(s)) d\mu(s) = 0,$$

and, therefore,

$$G(X(t)) \leq 0 \quad \text{a.e. on } [0, T],$$

but due to the continuity of G and right continuity of X , this inequality will be valid for any $t \in [0, T]$.

As follows from second relation nonnegative measure $\mu(dt)$ be localized on the set $\{t : G^-(X(t)) = 0\}$, however, due to the previous conclusion this set coincides with the set $\{t : G(X(t)) = 0\}$.

Remark 4. This theorem gives a representation of generalized solution of hybrid system (1), which corresponds to cases when the number of jumps could be equal to infinity and/or the sliding mode along the boundary $G(x) = 0$ could arise. Both cases are determined by the properties of measure $\mu(dt)$, i.e., the case $\mu^c([0, T]) > 0$ corresponds to the case of the sliding mode existence, and the case of infinite number of the atomic points of $\mu(dt)$ corresponds to the case of the infinite number of jumps.

Remark 5. The problem of correspondence between ordinary and generalized (relaxed) solution in presence either the sliding modes or the infinite number of jumps is non trivial. Generally it is not possible to approximate the generalized solution by a sequence of ordinary ones without constraints violation. However, it is possible to guarantee that this violation will be infirmly small and goes to zero, while the approximation sequence converges to generalized solution uniformly [7].

4 Existence of the optimal solution

So we come to the concept of generalized solution of hybrid system, which can be defined as a right continuous function $X(\cdot)$, such that $G(X(t)) \leq 0$, and satisfying the equation (18) with some admissible control u and nonnegative measure $\mu(dt)$, localized on the set $\{G(X(t)) = 0\}$.

As follows from previous results it make sense to search the solution of the original optimization problem in the class of generalized solutions. From Theorem 2 one can obtain the equivalence of the original optimization problem, which contains measures, to some auxiliary problem of nonsmooth optimization.

Auxiliary Problem. Consider the optimal control problem for system (13) with controls $\{\alpha, u_1\}$, satisfying (16), and such that the integral constraints (15) are valid. We will consider this problem on nonfixed time interval $[0, T_1]$, such that $T_1 < \infty : \eta(T_1) = T$ with performance criterion

$$J'\{y(\cdot), \alpha(\cdot), u_1(\cdot)\} = \phi_0(y(T_1)) \rightarrow \min,$$

where ϕ_0 is the same as in (8).

Theorem 3. Suppose that the set $F(X, U)$ be convex for any $X \in R^n$, the set of admissible controls of auxiliary problem is non empty, and the set of admissible T_1 , such that $\eta(T_1) = T$ is uniformly bounded. Then the auxiliary problem has the optimal solution and the optimal generalized solution of the original problem satisfies the equation (18) with appropriate control $u(\cdot)$ and measure $\mu(dt)$.

Proof. For any control $u(\cdot)$ and measure $\mu(dt)$, which give some generalized solution, one can define the appropriate controls $\{\alpha(s), u_1(s)\}$, which are admissible in auxiliary problem. Indeed, if

$$\Gamma(t) = t + \mu((0, t]),$$

and

$$\eta(s) = \inf\{t : \Gamma(t) \geq s\},$$

then the appropriate admissible controls $\{\alpha(s), u_1(s)\}$, can be defined by relations

$$\alpha(s) = \dot{\eta}(s), \quad u_1(s) = u(\eta(s)),$$

and

$$J[X(\cdot), u(\cdot)] = J'\{y(\cdot), \alpha(\cdot), u_1(\cdot)\}$$

since $X(T) = y(T_1)$. Therefore,

$$\inf_{u(\cdot)} J \geq \inf_{\{\alpha(\cdot), u_1(\cdot)\}} J'.$$

Due to the convexity assumptions and boundedness of T_1 the set of admissible paths of auxiliary problems be compact (see [4]), therefore the optimal control exists and the infimum in the right-hand side of above relation can be achieved on some controls $\{\alpha^0, u_1^0\}$.

To prove the existence of optimal solution for original problem it is sufficient to apply Theorem 2. Indeed, if $\{y^0, \alpha^0, u_1^0\}$ be the optimal solution of the auxiliary problem, then one can define $\{X^0, u^0, \mu^0\}$, such that $X^0(t) = y^0(\Gamma(t))$ and by virtue of conditions (15) we have

$$\int_0^T G^+(X^0(t))dt = \int_0^{T_1} \alpha^0(s)G^+(y^0(s))ds = 0,$$

$$\int_0^T G^-(X^0(t))d\mu^0(t) = \int_0^{T_1} (1 - \alpha^0(s))G^-(y^0(s))ds = 0,$$

therefore, $G(X^0(t)) \leq 0$ almost everywhere on $[0, T]$ and measure $\mu^0(dt)$ be localized on the set $\{t : G(X^0(t)) = 0\}$. However, $X^0(\cdot)$ be a right continuous function, thus the constraint $G(X^0(t)) \leq 0$ be valid for all $t \in [0, T]$.

Since $X^0(T) = y^0(T_1)$,

$$J[X^0(\cdot), u^0(\cdot)] = J'\{y^0(\cdot), \alpha^0(\cdot), u_1^0(\cdot)\} = \inf J',$$

and the triple $\{X^0, u^0, \mu^0\}$ be the optimal solution of the original problem in the class of generalized solutions.

Remark 6. The auxiliary problem belongs to a class of nonsmooth optimization problems due to the nondifferentiability of functions G^+ and G^- . However, by applying the methods recently obtained for nonsmooth problems (see, for example [2]), it becomes possible to derive necessary optimality conditions in the maximum principle form and design the computational algorithms.

References

1. B. Brogliato, *Nonsmooth Impact Mechanics. Models, Dynamics and Control*, Lecture Notes in Control and Information Sciences, No 220. Springer-Verlag (1996).
2. F. H. Clarke, *Optimization and Nonsmooth Analysis*, John Wiley & Sons, New York, Chichester, Brisbane, Toronto, Singapore (1983).

3. M. Jean and J. J. Moreau, *Dynamics of elastic or rigid bodies with frictional contact: numerical methods* Publications of Laboratory of Mechanics and Acoustics, Marseille, April, No 124, (1991).
4. E. B. Lee and L. Markus, *Foundations of Optimal Control Theory* John Wiley and Sons, Inc., New York, London, Sydney (1967).
5. B. M. Miller, "Method of Discontinuous Time Change in Problems of Control for Impulse and Discrete-Continuous Systems," *Autom. Rem. Control*, **54** (1993) 1727-1750.
6. B. M. Miller., "The generalized solutions of nonlinear optimization problems with impulse controls," *SIAM J. Control Optim.*, **34**, No. 4, 1420-1440 (1996).
7. B. M. Miller., "The generalized solutions of ordinary differential equations in the impulse control problems," *Journal of Mathematical Systems, Estimation, and Control*, **6**, No 4, 415-435, (1996).
8. B. M. Miller, "Representation of robust and non-robust solutions of nonlinear discrete-continuous systems" in Proceedings of International Workshop on Hybrid and Real-Time Systems (HART'97), Grenoble, France, March 26-28, (1997).
9. J. J. Moreau, "Unilateral contacts and dry friction in finite freedom dynamics", in Nonsmooth Mechanics and Applications, CIMS Course and Lectures, No 302, Springer-Verlag, Wien. New York, pp; 1-82, (1988).
10. Yu. V. Orlov, *Theory of Optimal Systems with Generalized Controls*. [in Russian], Nauka, Moscow (1988).
11. A. Ph. Phillipov, *Differential Equations with Discontinuous Right-Hand-Side*. [in Russian], Nauka, Moscow (1985).
12. L. C. Young, *Lectures on Variational Calculus and the Theory of Optimal Control*, W. B. Saunders Company, Philadelphia, London, Toronto, 1969.
13. S. T. Zavalishchin and A. N. Sesekin, *Impulsive Processes. Models and Applications* [in Russian], Nauka, Moscow (1991).

Information-Based Optimization Approaches to Dynamical System Safety Verification

Todd W. Neller*

Knowledge Systems Laboratory, Stanford University
e-mail: neller@ksl.stanford.edu

Abstract. Given a heuristic estimate of the relative safety of a hybrid dynamical system trajectory, we transform the initial safety problem for dynamical systems into a global optimization problem. We introduce MLL0-IQ and MLL0-RIQ, two new information-based optimization algorithms. After demonstrating their strengths and weaknesses, we describe the class of problems for which different optimization methods are best-suited.

The transformation of an initial safety problem for dynamical systems into a global optimization problem is accomplished through construction of a heuristic function which simulates a system trajectory and returns a heuristic evaluation of the relative safety of that trajectory. Since each heuristic function evaluation may be computationally expensive, it becomes desirable to invest more computational effort in intelligent use of function evaluation information to reduce the average number of evaluations needed. To this end, we've developed MLL0-IQ and MLL0-RIQ, information-based methods which approximate optimal optimization decision procedures.

* This work was supported by the Defense Advanced Research Projects Agency and the National Institute of Standards and Technology under Cooperative Agreement 70NANB6H0075, "Model-Based Support of Distributed Collaborative Design". Author's address: Knowledge Systems Laboratory, Gates Building 2A, Stanford University, Stanford CA 94305-9020.

1 Introduction

Given a simulated hybrid dynamical system S , a set of possible initial states I , and a set of "unsafe" states U , we wish to verify nonexistence of an S -trajectory from I to U within t_{\max} time units. We call this the *initial safety problem*. Suppose we are given an approximate measure of the relative safety of a trajectory. More specifically, let f be a function taking an initial state i as input, and evaluating the S trajectory from i such that $f(i) = 0$ if and only if the S -trajectory from i enters U within t_{\max} time units, and $f(i) > 0$ otherwise. Then verification of the initial safety problem can be transformed into the global optimization (GO) problem:

$$\min_{i \in I} (f(i)) \stackrel{?}{>} 0$$

GO methods may therefore terminate when i is found such that $f(i) = 0$. Given that f does not generally have an analytic form, we do not assume the availability of derivatives. Since each evaluation of f may require a computationally expensive simulation, we are particularly interested in GO methods which perform relatively few evaluations of f . In this context, we introduce two new information-based optimization methods which use function evaluation information approximately optimally in choosing the next best point for evaluation. We demonstrate that these algorithms generally match or exceed the performance of the best methods from our previous comparative study [1], describe the class of functions for which they are best suited, and finally turn our attention to the trade-off between brute-force function evaluation and intelligent, selective function evaluation.

2 Motivation

Our research was largely motivated by the following safety verification task: Given bounds on the system parameters of a stepper motor (e.g. viscous friction, inertial load), bounds on initial conditions (e.g. angular displacement and velocity), and an open-loop motor acceleration control, verify that no scenario exists in which the motor stalls. We model the motor's continuous dynamics using ODEs given in [2]:

$$\begin{aligned}\dot{\theta} &= \omega \\ \dot{\omega} &= \frac{(-i_a N_b \sin(N\theta) + i_b N_b \cos(N\theta) - D \sin(4N\theta) - F_v \omega - F_c \text{sign}(\omega) - F_g)}{(J_l + J_m)} \\ i_a &= (V_a - i_a R + \omega N_b \sin(N\theta))/L \\ i_b &= (V_b - i_b R - \omega N_b \cos(N\theta))/L\end{aligned}$$

where θ and ω are motor shaft angular displacement and velocity, i_a and i_b are coil A and B current, V_a and V_b are coil A and B voltage, R and L are coil resistance and inductance, N is the number of rotor teeth, N_b is the maximum

motor torque per amp, D is the maximum detent torque, F_v is the viscous friction, F_c is the Coulomb friction, F_g is the gravitational torque load, and J_l and J_m are load and motor shaft inertia. For this system we classify a stall as deviation of $\frac{\pi}{N}$ or more radians from the current desired θ equilibrium.

The motor is stepped by reversing polarity of the coil voltages in alternation (see Figure 1). Changes to coil voltages occur on such a small time scale that their continuous simulation is judged unnecessary for modeling dynamics relevant to the verification task. Voltage changes were therefore approximated as discrete events. Our acceleration control is open-loop: At fixed intervals the motor is stepped according to an acceleration table. We can express such a system as a nonlinear hybrid automaton as shown in Figure 2.

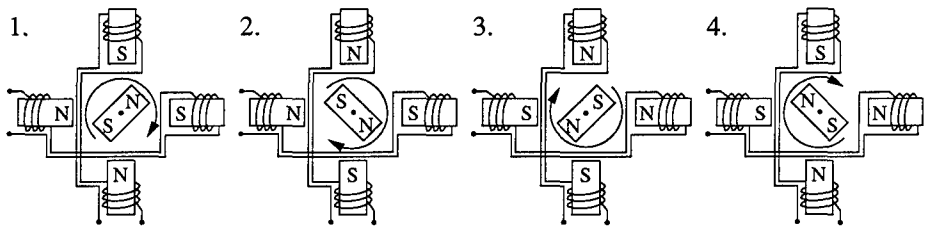


Fig. 1. Simple Stepper Motor Stepping

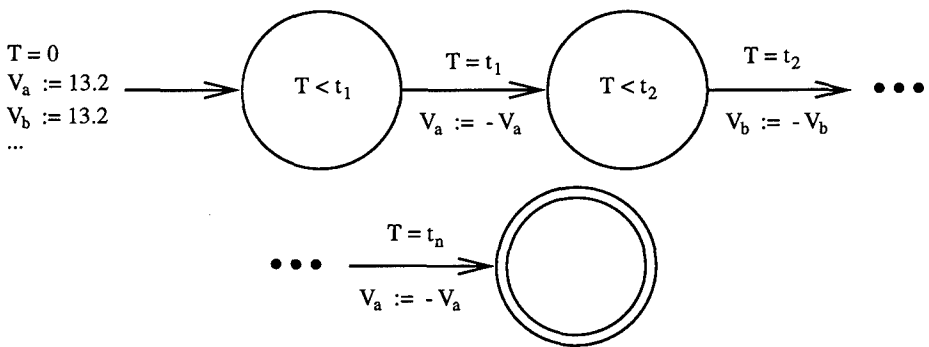


Fig. 2. Stepper Motor Nonlinear Hybrid Automaton

First, we note that there is no apparent “geometrically linear hybrid system”¹ approximation with which we could apply the tools of computational geometry, but simulation is feasible. Next, we note that our verification is concerned with

¹ i.e. restricted to constant first derivatives; “geometrically” as opposed to “algebraically”

a fixed initial time interval (i.e. during acceleration) and is therefore an initial safety problem. Finally, we note that we can compute minimum angular displacement from a stall state over all simulation states as a simple heuristic to numerically rate the relative safety of safe trajectories. We can now ask, "For all possible system parameters and initial states, are all simulation trajectories rated safe?" Put another way, "Is the minimum heuristic evaluation of all possible simulations greater than zero?" If we can answer this optimization question positively, we have verified safety of our hybrid system.

One could argue that such optimization is *not* verification, that one cannot exhaustively simulate all possibilities and can therefore have no guarantees. One can only use such optimization for refutation. To this, we offer two responses: First, if one has additional knowledge of characteristics of one's heuristic evaluation function, then an intelligent optimization approach can utilize such characteristics to guarantee a strictly positive minimum (i.e. safety) with enough testing. For example, if one is seeking a zero minimum of a heuristic function which has Lipschitz conditions, and there is no possibility for a zero to occur between previously evaluated points without violating such conditions, one can terminate the optimization having verified safety. Second, if one has no such knowledge about the heuristic (as is the case for our stepper motor problem), the absence of verification techniques well-suited to non-trivial dynamics leaves good global optimization as the best assurance. As has been demonstrated with several NP-hard satisfiability problems [3], refutation through a well-chosen optimization technique, while not complete, can open the door to solving larger classes of problems reliably.

This said, we have endeavored to develop innovative information-based global optimization methods which, under certain assumptions and constraints, make approximately optimal use of information gained in the course of optimization. We next introduce some of these methods.

3 Information-Based Global Optimization

From the previous comparative study [1], we noted that most global optimization methods throw away most of the information gained in the course of optimization. For our purposes, each evaluation of f requires a simulation which may be computationally expensive, so we are particularly motivated to make good use of such information in order to reduce the function evaluations needed.

One approach is to characterize properties of the set of functions one wishes to optimize and to use such information to construct an optimal decision procedure for optimization. In the course of optimization, we use our current set of function evaluations to decide on the next best point to evaluate with respect to our function set. Such is the strategy of *Bayesian* or *information* approaches to global optimization [4–7], which have optimal average case behavior over the set of functions for which each is designed. Previous information-based methods have largely been limited to global optimization in one dimension. In this section, we

introduce two new information-based optimization methods for multidimensional problems.

We first introduce the decision procedure used by these methods, thus explicating the class of functions for which the decision procedure is biased. Next we discuss the use of *multi-level local optimization* for speeding convergence. Finally, we introduce the information-based optimization algorithms themselves.

3.1 Decision Procedure

At each iteration i of our algorithm, we wish to evaluate our heuristic function f at the location x_i for which $f(x_i) = 0$ is most likely to occur. We base our notion of likelihood on characteristics of a class of functions to which f belongs. Our decision procedure is then based on some decision ranking function g_i which computes a ranking corresponding to the relative likelihood of a zero occurring at an unevaluated point x_i given previous f -evaluations at x_1, x_2, \dots, x_{i-1} :

$$g_i(x_i) \stackrel{\text{def}}{=} g(x_1, x_2, \dots, x_{i-1}, x_i)$$

So for each iteration i , we could globally optimize g_i to choose the next x for which f is evaluated. However, a reliable global optimization of g for each iteration of a global optimization of f is not only computationally prohibitive, but increasingly very difficult as well. We instead desire to *approximate* an optimal decision with respect to our assumptions about f , and we do so by uniformly, randomly sampling g , returning the optimum of the samples. We call this DECISION1 (see Function 1). The computational complexity of this decision procedure grows as the computational complexity of evaluating g_i (which we will see is $O(i^2)$).

Function 1 Sampling information-based optimization decision function

```

DECISION1(L,lbound,ubound):
% Input: L, a list of [x,f(x)] pairs
%         lbound, lower bounding corner of search space
%         ubound, upper bounding corner of search space

mingx := infinity
for i := 1 to maxpts
  x := uniformly random vector in space bounded by lbound and ubound
  gx := g(L,x)
  if gx < mingx then
    mingx := gx
    minx := x
  end
end for
return minx

```

In order to construct g , we must make some assumptions over f 's class of functions with regard to where we would most expect to find zeros. One assump-

tion we make is that f is continuous². Another assumption concerns flatness and smoothness preferences: Given a set of points and their f -evaluations, a zero is more likely to occur where it demands less slope between itself and previous points.

A first attempt at constructing g_i might be to create a function which returns

$$g_i(x) = \max_{j=1}^{i-1} \frac{f(x_j)}{\|x_j - x\|}$$

That is, we could rank the likelihood of $f(x) = 0$ by computing the maximum slope between the hypothetical zero at x and other points we've already evaluated. The lesser the g -value, the more likely a zero f -value. The global minimum of g would then be the optimal point at which to next evaluate f given previous f evaluations.

Consider Figure 3(a). Suppose we've evaluated the curve at points a , b , and c and are using such a g as our decision ranking function. Intuitively, we would want g to return point d as the next best point to evaluate. However, the slope between a and d will make d a less preferable decision point than one to the right of d for which a zero would have equal slopes to a and c for this simple function. We would like instead for point b to "shadow" point d from point a . Our simple attempt to do so is shown as Function 2. A point a is "shadowed" by point b for function g if $\|d - b\| < \|d - a\|$ and $|g(a) - g(b)| / \|a - b\| > |g(a) - g(d)| / \|a - d\|$. That is, a is shadowed by b if b is closer to d than a , and the slope between a and b on g is greater than the slope between a and d on g .

3.2 Multi-Level Local Optimization

One might then construct the simple information-based global optimization procedure given in Program 1. However, we note that one ramification of random sampling in our decision procedure is that we do not achieve efficient convergence. This is illustrated in Figure 3(b). From the initial random point in the lower left corner, the procedure then checks points in the upper right, lower right, upper left, and just left of the global minimum at the center. The cluster of 25 points that follows gradually expands towards the center from the fifth point. In practice, where failures do not occur in miniscule regions, this behavior is not a problem. However, we also note that our decision procedure will have to deal with the computational burden of small dense clusters of points which are not very informative globally. We may wish instead to apply a rapidly convergent local optimization procedure and pay attention only to the first and last points of such an optimization.

In our previous comparative study [1], we note that this is a common approach among the most successful methods of the study. A global search phase

² This is not a trivial assumption for our general application, of course. Our stepper motor system trajectories are continuous in the initial condition. Such continuity is preserved in our choice of f .

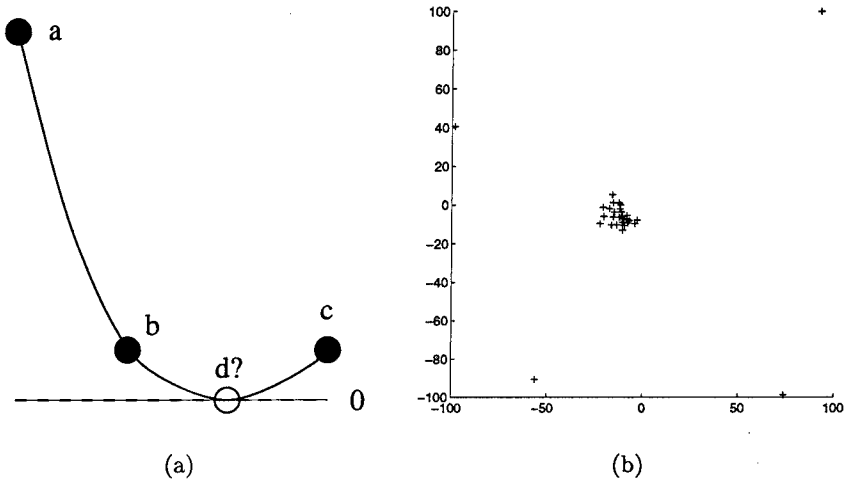


Fig. 3. (a) Shadowing example, (b) Information-based global optimization of 2-D paraboloid

makes use of a local optimization subroutine so that the global phase is, in effect, searching $f'(x_1) \stackrel{\text{def}}{=} f(x_2)$ where $[x_2, f_{\min}] = \text{LO}(f, x_1)$ where LO is a local optimization procedure. In SALO [8] (simulated annealing atop local optimization), for each point evaluation in the global phase, a local optimization takes place and the function value of the local minimum is associated with the original point. The effect can be roughly described as a “flattening” of a search space into many plateaux (with plateaux corresponding to local minimum values). This search paradigm may be generalized to arbitrary levels where each level performs some optimizing transformation of its search landscape to create a “simpler” one for the level above. Obviously, the work done to simplify should be more than compensated by the reduced search effort for the level above. The top level performs a global optimization, and all lower levels perform local optimization. We call this paradigm *Multi-Level Local Optimization* (MLLO). We assert that information-based optimization is particularly well-suited to optimizing coarsely plateaued search landscapes. Now let us consider two information-based applications of MLLO.

3.3 MLLO-IQ and MLLO-RIQ

MLLO-IQ (Program 2) is a 2-level MLLO with a simple information-based approach (Program 1) atop quasi-Newton local optimization. With each iteration, MLLO-IQ chooses a point x_1 , locally optimizes f from x_1 to x_2 , and associates $f(x_2)$ with both x_1 and x_2 in order to “plateau” the space. In doing so, we limit the number of function values involved in decision making. Still, we may wish to further limit such growth in computational complexity. By limiting our

Function 2 *g*, the decision procedure function to be optimized

```

g(L,x):
% Input: L, a list of [x,f(x)] pairs
%       x, current decision point being evaluated

for i := 1 to length(L)
  dx(i) := ||x-first(L(i))||
end for
sort dx in ascending order and permute L accordingly
maxslope := 0
for i := 1 to length(L)
  slope := second(L(i))/dx(i)
  if slope > maxslope then
    newmaxslope := 1
    for j := 1 to i-1
      otherslope := |second(L(i))-second(L(j))|
                  /||first(L(i))-first(L(j))||
      % Note: This otherslope information may be cached.
      if otherslope > slope then
        newmaxslope := 0; break from for loop (j)
      end for
    if newmaxslope then maxslope := slope
  end if
end for
return maxslope

```

information-based search to a hypersphere containing a maximum limit of previously evaluated points, we limit the complexity to a constant. Such is the approach taken in MLO-RIQ.

MLO-RIQ (see Program 3) begins with a locally minimized random point and a maximum search radius. Together these define our initial hypersphere. With each iteration, a decision procedure (DECISION2) finds an approximately optimal next point to locally optimize within this hypersphere. If the new point has a lesser function value than the center, it becomes the new center and the distance between the two points becomes the new hypersphere radius. If too many points are being considered in DECISION2, a lesser amount of points closest to center are retained and the search radius is adjusted. This information-based local optimization terminates when the number of times the center minimum is found by local optimization exceeds a threshold. Then the process repeats with a new random point. Thus we perform a random search of information-based local optimizations of quasi-Newton local optimizations.

Program 1 Simple information-based global optimization

```

H = [];
newx := random point in search space
fx := f(newx)
if fx = 0 then terminate with signal UNSAFE
H := append(H,[newx,fx])
loop forever
  newx := DECISION1(H,lbound,ubound)
  fx := f(newx)
  if fx = 0 then terminate with signal UNSAFE
  H := append(H,[newx,fx])
end loop

```

Program 2 MLLO-IQ

```

H = [];
newx1 := random point in search space
[newx2,fx] := LO(f,newx1)
if fx = 0 then terminate with signal UNSAFE
H := concatenate(H,[[newx1,fx],[newx2,fx]])
loop forever
  newx1 := DECISION1(H,lbound,ubound)
  [newx2,fx] := LO(f,newx1)
  if fx = 0 then terminate with signal UNSAFE
  H := concatenate(H,[[newx1,fx],[newx2,fx]])
end loop

```

4 Experimental Results

We now compare our information-based approaches to those considered in our previous comparative study. See [1] for details and references. Our first tests all made use of the same quasi-Newton local optimization method where applicable. 100 optimization trials were performed for each objective function with a maximum of 10000 function evaluations permitted per trial. Each objective function was offset (if necessary) to have a global minimum value of 0. A successful trial was one in which the optimization procedure found a point with function value less than .001 within 10000 function evaluations. This simulates situations where one is seeking a rare failure case in f . Each entry in the table of results (Figure 4) shows the number of successful trials (upper left) and the average number of function evaluations for such trials (lower right) for each optimization procedure (rows) and objective function (columns).

Both MLLO-IQ and MLLO-RIQ perform very well in general. What is most instructive from these results are the cases where the strengths and weaknesses of these methods are most prominently displayed. Let us first consider RAST, the Rastrigin function. RAST is a 2-D, sinusoidally-modulated, shallow paraboloid

Program 3 MLLO-RIQ

```

H = []; radius := maxradius
loop forever
  x := random point in search space
  [center,centerval] := LO(f,x)
  if centerval = 0 then terminate with signal UNSAFE
  H := concatenate(H,[[x,centerval],[center,centerval]])
  sort pairs in H in ascending order of ||first(pair)-center||
  H' := up to first (minpts) pairs of H
  centerhits := 0
  while centerhits > maxcenterhits
    recenter := false
    newx1 := DECISION2(H',center,radius)
    [newx2,fx] := LO(f,newx1)
    if fx = 0 then terminate with signal UNSAFE
    if ||newx2-center|| < tolerancel then
      centerhits := centerhits + 1
    if centerval - fx > tolerance2 then
      radius := min(maxradius, ||newx2-center||)
      center := newx2; centerval := fx; centerhits := 0; recenter := true
    H := concatenate(H,[[newx1,fx],[newx2,fx]])
    H' := concatenate(H,[[newx1,fx],[newx2,fx]])
    if length(H') > maxpts then
      recenter := true
    if recenter then
      sort pairs in H in ascending order of ||first(pair)-center||
      H' := up to first (minpts) pairs of H
  end while
end loop

```

with 49 local minima within the search bounds. The quasi-Newton local optimization layer of MLLO-IQ and MLLO-RIQ effectively transforms this objective function f into a shallow paraboloid of plateaux f' . MLLO-IQ's global information-based search of f' finds the lowest plateau very quickly, and the local information-based search of MLLO-RIQ does a focussed descent which leads it to the global minimum with even greater efficiency. This suggests that these searches are particularly well-suited to global optimization of functions with a moderate number of local minima. For functions with fewer local minima (HUMP, G-P, and GW1), there is little to be gained by such extra computation. Random local optimization (RANDLO) will suffice.

Now let us consider the weaknesses of these methods shown in failed cases with GW100. Indeed the performance of these methods is worse than random local optimization. Why? GW100 is a 6-D, sinusoidally-modulated, shallow paraboloid with about 4×10^7 local minima. For this function, our quasi-Newton local optimization exhibits interesting and unexpected behavior: In all but the lowest points of the surface, local optimization most often leads to local minima that

	RAST	HUMP	G-P	GW1	GW100	SWISS
AMEBSA	16 39	100 40	90 222	100 86	0 N/A	100 1340
ASA	100 404	100 225	100 1042	100 197	2 6003	100 903
SALO	100 585	100 65	100 97	100 85	95 4501	100 163
LMSL	100 847	100 105	100 118	100 96	50 4508	100 187
RANDLO	100 706	100 70	100 96	100 85	58 4008	100 146
MLLO-IQ	100 286	100 71	100 97	100 83	57 4493	100 157
MLLO-RIQ	100 161	100 57	100 92	100 83	46 4536	100 148

Fig. 4. Successful trials and average function evaluations for each global optimization procedure and test function

are far from those nearby the initial point. In this example, we're reminded that "local" in "local optimization" refers to properties of the optimum itself and not the "nearness" of the optimum location. Without such nearness, the search landscape is not simply transformed into a landscape of plateaux. Our quasi-Newton local optimization didn't optimize to near minima, and so created a landscape which was not suited for information-based global optimization.

MLLO-RIQ also has difficulty with GW100, but for different reasons. After quickly finding the region containing the global minimum, the method spends much of the remainder of its search effort first searching many points mutually far apart near the boundary of the 6-D hypersphere. Perhaps randomly sampling f or f' within the search hypersphere might encourage convergence. SALO remains our best option for functions with a large number of local minima.

While these functions may give a general indication of the relative strengths of these methods (without tuning), the functions share a common property undesirable for our purposes: The *unconstrained* global minimum is never located at or beyond the bounds of the search space. Therefore, our optimization methods need not perform well along the bounds of our search space. It is for this reason that unconstrained quasi-Newton local optimization was suitable for use with such global optimizations. We used this as an opportunity to try two constrained LO procedures CONSTR and YURETMIN for the 2-D stepper motor test problems STEP1 and STEP2 [1] (see Figure 5). STEP1 takes as input two parameters (viscous friction and load inertia) of the stepper motor model, simulates acceleration of the motor, and performs a simple heuristic evaluation of the trajectory by computing the minimum distance to a stall state (0 if stalled). One could incorporate more sophisticated understanding of a problem domain into one's heuristic function, but computing the minimum distance to an undesirable state is simple and effective for our purposes. STEP2 is STEP1 logarithmically

scaled so as to focus on the unsafe region of the parameter space. These test functions were chosen for their difficulty. For this testing, we performed 10 trials to find a function value of 0 with a maximum of 1000 function evaluations per trial. The results appear in the tables of figure 6.

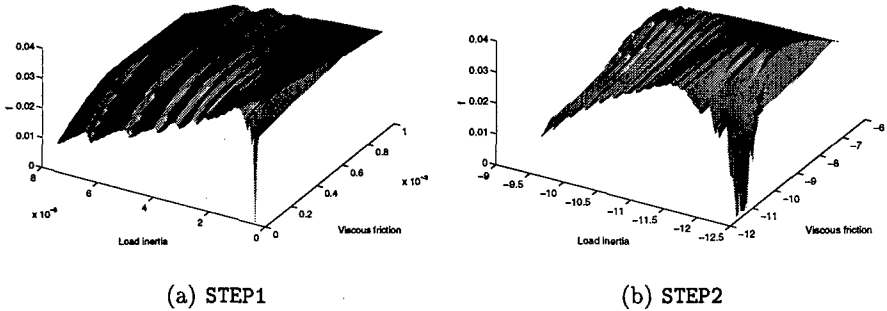


Fig. 5. Stepper Motor Test Functions

These results were very pleasing. MLLO-IQ is the first technique we've observed that has succeeded in every STEP1 and STEP2 trial. It does so with excellent efficiency as well. Since the decision procedure computation time was also dominated by simulation time, it was also easily the fastest algorithm for these trials. MLLO-RIQ did surprisingly well considering that most of the search space of these functions slopes downward and away from the corner of the space where the rare failure cases occur.

5 Conclusions

A powerful approach to initial safety verification is to transform the problem into an optimization problem and leverage the power of efficient optimization methods. This transformation is accomplished through a heuristic evaluation function f which takes an initial state as input, simulates the corresponding trajectory, and evaluates the trajectory, returning zero if the trajectory is unsafe, or a strictly positive ranking of the relative safety of the trajectory otherwise. Initial safety verification is then a matter of whether or not the global minimum of f for all possible initial states is strictly positive. Our simple heuristic function computes the minimum distance from a trajectory to an unsafe state, but deeper understanding of the problem domain may be incorporated as well.

Although we have not investigated the applicability of optimization to non-deterministic hybrid systems, we believe such techniques are applicable to a

	STEP1	STEP2		STEP1	STEP2
ASA	0 N/A	2 497	ASA	0 N/A	2 497
SALO	10 80	5 202	SALO	7 387	9 198
LMLSL	10 163	10 137	LMLSL	3 389	10 169
RANDLO	10 78	10 359	RANDLO	9 501	10 172
MONTE	0 N/A	6 469	MONTE	0 N/A	6 469
MLLO-IQ	10 46	10 219	MLLO-IQ	10 108	10 109
MLLO-RIQ	10 60	8 330	MLLO-RIQ	8 301	9 239

(a) CONSTR

(b) YURETMIN

Fig. 6. Results for STEP1 and STEP2

broader class of deterministic hybrid systems than we have demonstrated. Use of problem domain knowledge to construct a heuristic function and choose global and local optimization techniques should expand the frontier of solvable hybrid system problems. Optimization techniques which are robust with respect to discontinuities should be used for most hybrid system initial safety problems.

From the previous comparative study [1], we noted that most global optimization methods throw away most of the information gained in the course of optimization. For our purposes, each evaluation of f requires a simulation which may be computationally expensive, so we are particularly motivated to make good use of such information in order to reduce the function evaluations needed. To this end, we have introduced two new information-based global optimization methods MLLO-IQ and MLLO-RIQ which, under certain assumptions and constraints, make approximately optimal use of information gained in the course of optimization. Our decision procedure is biased towards approximately optimal average-case behavior for a subclass of continuous heuristic functions.

While no global optimization procedure in our studies was generally dominant, we note that random local optimization seems best suited for heuristic functions with few minima, SALO [8] seems best suited for heuristic functions with very many local minima, and MLLO-IQ and MLLO-RIQ seem best suited for low-dimensional heuristic functions with a moderate number of local minima. MLLO-IQ and MLLO-RIQ appear better suited for problems where the global minima are expected to occur at and within the bounds of the search space respectively.

Finally, we note that the computational effort invested toward efficient optimization should be compensated by reduced overall runtime. For our prob-

lem, the computational expense of our simulation justified such effort. But what of initial safety problems for which simulation requires less runtime? Setting $\text{maxpts} = 0$ for Function 1 yields random decisions. As $\text{maxpts} \rightarrow \infty$, our decisions approach optimality and the decision-making effort exceeds the search effort it saves. Where is the happy medium in this tradeoff? In future research, we hope to investigate means of dynamically adjusting the level of strategic effort of such information-based algorithms in order to address a larger class of problems efficiently.

References

1. Todd W. Neller. Heuristic optimization and dynamical system safety verification. In M. Lemmon, editor, *Proceedings of Hybrid Systems V (HS '97)*, pages 51–59, South Bend, IN, USA, 1997. Center for Continuing Education, University of Notre Dame. (available at URL <http://www.ksl.stanford.edu/people/neller/pubs.html>).
2. Albert C. Leenhouts. *Step Motor System Design Handbook*. Litchfield Engineering, Kingman, Arizona, USA, 1991.
3. Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 440–446, Menlo Park, CA, USA, 1992. AAAI Press.
4. Jonas Mockus. *Bayesian Approach to Global Optimization: theory and applications*. Kluwer Academic, Dordrecht, The Netherlands, 1989.
5. Jonas Mockus. Application of bayesian approach to numerical methods of global and stochastic optimization. *J. Global Optimization*, 4:347–365, 1994.
6. Yaroslav D. Sergeyev. An information global optimization algorithm with local tuning. *SIAM J. Optimization*, 5(4):858–870, 1995.
7. Roman G. Strongin. The information approach to multiextremal optimization problems. *Stochastics and Stochastics Reports*, 27:65–82, 1989.
8. Rutvik Desai and Rajendra Patil. SALO: combining simulated annealing and local optimization for efficient global optimization. In J.H. Stewman, editor, *Proceedings of the 9th Florida AI Research Symposium (FLAIRS-'96)*, pages 233–237, St. Petersburg, FL, USA, 1996. Eckerd Coll.

Synthesizing Controllers for Nonlinear Hybrid Systems*

Claire Tomlin, John Lygeros, and Shankar Sastry

Department of Electrical Engineering and Computer Sciences
University of California,
Berkeley CA 94720
clairet, lygeros, sastry@eecs.berkeley.edu

Abstract. Motivated by an example from aircraft conflict resolution we seek a methodology for synthesizing controllers for nonlinear hybrid automata. We first show how game theoretic methodologies developed for this purpose for finite automata and continuous systems can be cast in a unified framework. We then present a conceptual algorithm for extending them to the hybrid setting. We conclude with a discussion of computational issues.

1 Introduction

In the first part of this paper we present a motivating example: we describe an iteration process to calculate the maximal set of safe initial conditions for a two-aircraft maneuver. In the second part we show that verification of the safety of continuous nonlinear systems using the Hamilton-Jacobi-Bellman equation may be considered as the continuous analog of infinite games on finite automata. In the third part we present a conceptual algorithm for calculating maximal controlled invariant sets for nonlinear hybrid systems, and we conclude with a brief discussion of computational issues.

The idea of posing the controller synthesis problem as a discrete game between the system and its environment is attributed to Church [1], who was studying solutions to digital circuits. The solution to this problem using a version of the von Neumann-Morgenstern discrete game [2] is due to Büchi and Landweber [3] and Rabin [4]. [5] also discusses games on automata. A comprehensive modern survey of infinite discrete games on automata is presented in [6] and [7]. Controller synthesis on timed automata was first developed in [8] and [9]. An algorithm for controller synthesis on linear hybrid automata is presented in [10]. The notion of control invariance for continuous systems is described in [11], and control invariance for hybrid systems is discussed in [12].

The study of differential equations in game theory was first motivated by military problems in the U.S. Air Force (aircraft dog fights, target missiles) and was initially developed by Isaacs in the 1940's and 50's [13]. An excellent modern

* Research supported by NASA under grant NAG 2-1039, by the California PATH program under MOU-238 and MOU-288, and by a Zonta Postgraduate Fellowship.

reference is [14]. Our motivation for this work arose out of attempting to verify the safety of a class of conflict resolution maneuvers for aircraft, in [15]. Similar previous work is that of [16], in which game theoretic methods were used to prove safety of a set of maneuvers for Automated Highway Systems.

Let us first introduce some basic notation. Let PC^0 denote the space of piecewise continuous functions over \mathbb{R} , and PC^1 the space of piecewise differentiable functions over \mathbb{R} .

Entity	Discrete	Continuous
State Space	Q	\mathbb{R}^n
Input Sets	$\Sigma_0 \times \Sigma_1$	$U \times D$
Input Space	$\Sigma_0^\omega \times \Sigma_1^\omega$	$\mathcal{U} \times \mathcal{D} \subset PC^0 \times PC^0$
Transition Relation	$\delta : Q \times \Sigma_0 \times \Sigma_1 \rightarrow 2^Q$	$f : \mathbb{R}^n \times \mathcal{U} \times \mathcal{D} \rightarrow \mathbb{R}^n$:
		$\forall \tau, \dot{x}(\tau) = f(x(\tau), u(\tau), d(\tau))$
System Trajectory	$(\gamma, s_0, s_1) \in Q^\omega \times \Sigma_0^\omega \times \Sigma_1^\omega$: $\gamma[i+1] \in \delta(\gamma[i], s_0[i], s_1[i])$	$(x(\cdot), u(\cdot), d(\cdot)) \in PC^1 \times \mathcal{U} \times \mathcal{D}$: $\forall \tau, \dot{x}(\tau) = f(x(\tau), u(\tau), d(\tau))$
Acceptance Conditions	$\Box F; \Diamond G$	$\forall \tau, x(\tau) \in F; \exists \tau, x(\tau) \in G$

2 Motivating Example

Consider a variation of the two aircraft collision avoidance problem of [15], in which there are two modes of operation: a *cruise* maneuver in which both aircraft follow a straight path, and an *avoid* maneuver in which both aircraft follow a circular arc path. The protocol of the maneuver is that as soon as the aircraft are within a distance α of each other, each aircraft turns 90° to its right and follows a half circle. Once the half circle is complete, each aircraft returns to its original heading and continues on its straight path (Figure 1). Safety is defined in terms of the relative distance between the two aircraft: throughout the maneuver the aircraft must remain at least 5 miles apart. In this section, we calculate the largest set of initial conditions of the system which render this maneuver safe, implicitly determining the parameter α in the process.

In each mode, the nonlinear dynamics may be expressed in terms of the *relative motion* of the two aircraft (equivalent to fixing the origin of the relative frame on aircraft 0 and studying the motion of aircraft 1 with respect to aircraft 0):

$$\begin{aligned}
 \dot{x}_r &= -v_0 + v_1 \cos \phi_r + \omega_0 y_r \\
 \dot{y}_r &= v_1 \sin \phi_r - \omega_0 x_r \\
 \dot{\phi}_r &= \omega_1 - \omega_0
 \end{aligned} \tag{1}$$

in which (x_r, y_r, ϕ_r) is the relative position and orientation of aircraft 1 with respect to aircraft 0, and v_i and ω_i are the linear and angular velocities of each aircraft. In mode 1, $\omega_i = 0$ for $i = 0, 1$ and in mode 2, $\omega_i = 1$ for $i = 0, 1$. The control input is defined to be the linear velocity of aircraft 0, $u = v_0 \in U$, and the

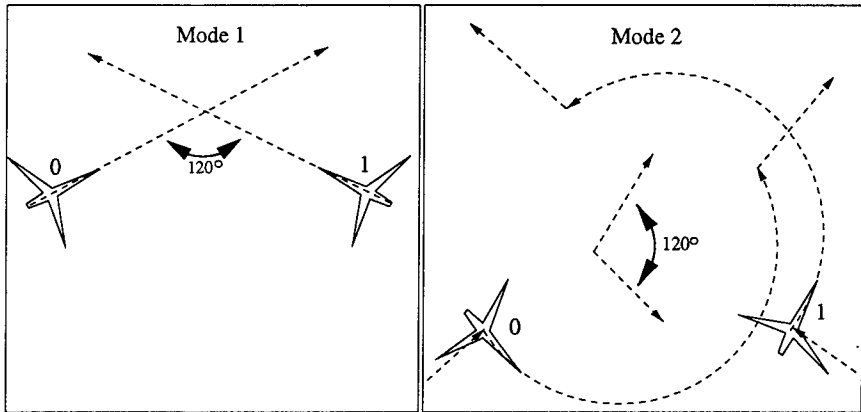


Fig. 1. Two aircraft in two modes of operation: in mode 1 the aircraft follow a straight course and in mode 2 the aircraft follow a half circle. The initial relative heading (120°) is preserved throughout.

disturbance input as that of aircraft 1, $d = v_1 \in D$, where U and D are called the control and disturbance sets and denote the range of possible linear velocities of each aircraft. Such a situation could arise, for example, in an airborne collision avoidance algorithm in which the flight management system of aircraft 0 wishes to compute the parameters v_0 and α of its avoidance maneuver and can only predict the velocity of aircraft 1 to within some uncertainty.

We define the region at which "loss of separation" occurs as a 5-mile-radius cylinder around the origin in the (x_r, y_r, ϕ_r) space:

$$G = \{(x_r, y_r) \in \mathbb{R}^2, \phi_r \in [-\pi, \pi) \mid x_r^2 + y_r^2 \leq 5^2\} \quad (2)$$

This region is referred to in the pursuit-evasion game literature as the *capture set*.

We now describe pictorially the calculation of the largest set of initial conditions $(x_r(0), y_r(0)) \in \mathbb{R}^2$ which render the maneuver safe. Consider the four consecutive plots of Figure 2. Aircraft 0 is at the origin of the relative axis, and G is the capture set. Since in both mode 1 and mode 2 the relative orientation between the two aircraft is constant ($\dot{\phi}_r = 0$), the value of ϕ_r acts as a parameter. We therefore consider the maneuver in the (x_r, y_r) plane.

In the first plot, the set V_1^* denotes the winning states for aircraft 1 in mode 1: those states from which for all possible actions of aircraft 0, aircraft 1 has an action which can drive it into G :

$$V_1^* = \{(x_r(0), y_r(0)) \mid \exists \tau \in [0, \infty), (x_r(\tau), y_r(\tau)) \in G\} \quad (3)$$

where $x(\tau)$ evolves according to the dynamics in mode 1.

Plot 2 illustrates V_2^* , the winning states for aircraft 1 in mode 2 with the stipulation that the aircraft remain in mode 2 for exactly π seconds (to complete

the half circle):

$$V_2^\pi = \{(x_r(0), y_r(0)) \mid \exists \tau \in [0, \pi], (x_r(\tau), y_r(\tau)) \in G\} \quad (4)$$

where $x(\tau)$ evolves according to the dynamics in mode 2.

Plot 3 illustrates the set V_1^* transformed into the relative frame for mode 2 (this *reset map* $r_1(\cdot)$ rotates every state in mode 1 by 90° , corresponding to aircraft 0 at the origin of the relative frame rotating by -90° when switching from mode 1 to mode 2). The intersection $r_1(V_1^*) \cap V_2^\pi$, shown as the darker shaded area of plot 3, represents those states which are potentially unsafe, since outside of this intersection, the aircraft may always switch modes to achieve safety. Plot 4 displays V^* , the minimal unsafe set. Note that the set of states $r_1(V_1^*) \cap V_2^\pi \setminus V^*$ has been removed from the unsafe set of states at this step of the iteration by flowing the dynamics of mode 1 forward in time, and then switching from mode 1 to mode 2 before the system enters G .

In the calculation of the minimal unsafe set of Figure 2, the control and disturbance sets U and D are singletons: both $u = v_0$ and $d = v_1$ are given. Thus in this example the *action* refers only to α , the minimum relative distance at which the aircraft must switch to mode 2, since we have fixed the velocities of the two aircraft at known values. In the remainder of the paper, we formalize this calculation for arbitrary control and disturbance sets, arbitrary nonlinear equations describing the continuous dynamics, and arbitrary invariant conditions for the discrete modes.

3 Verifying safety in continuous systems: a comparison with discrete \square - and \diamond -games

3.1 Infinite Games on Finite Automata

We summarize a class of two-player games on finite automata, in which the goal of Player 0 is to force the system to remain inside a certain “good” subset of the state space, and the goal of Player 1 is to force the system to leave this same subset. We describe the iteration process for calculating the set of states from which Player 0 can always win, and the set of states from which Player 1 can always win. We then show how this iteration process can be written as a difference equation for a *value* function, similar to the Hamilton-Jacobi-Bellman equation for differential games on continuous systems.

System Definition and Winning Condition We consider two players P_0 and P_1 , playing over a *game automaton* of the form:

$$(Q, \Sigma, \delta, Q_0, \Omega) \quad (5)$$

where Q is a finite set of *states*, Σ is a finite set of *actions*, $\delta : Q \times \Sigma \rightarrow 2^Q$ is a *partial transition relation*, $Q_0 \subseteq Q$ is a set of *initial states*, and Ω is a *trajectory acceptance condition*. The set of actions is the product of two sets $\Sigma = \Sigma_0 \times \Sigma_1$

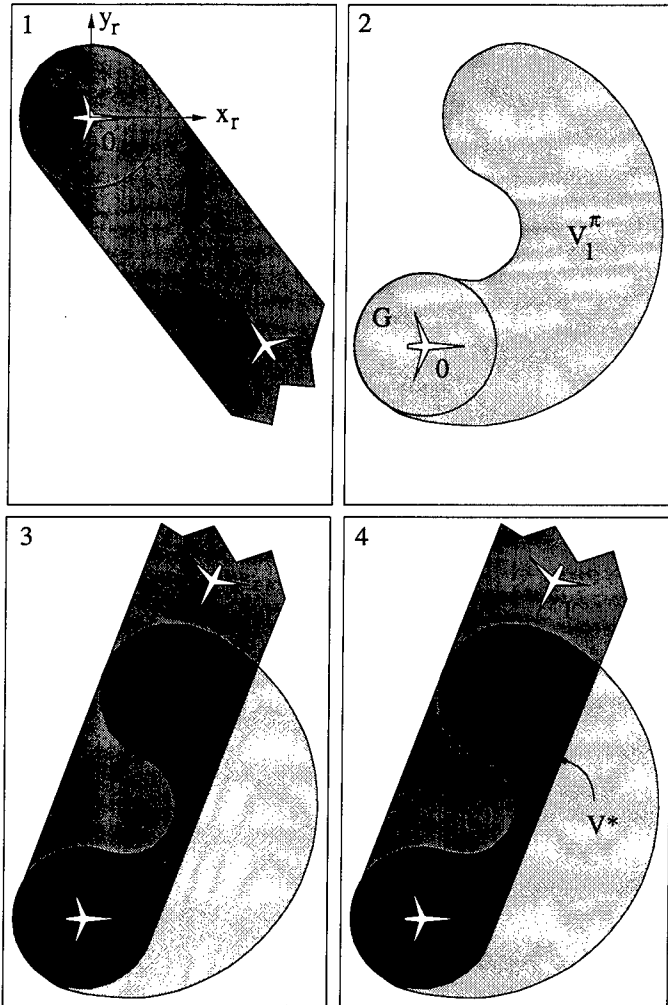


Fig. 2. Showing the successive calculation of the minimal unsafe set.

where Σ_i contains the action of P_i , so that each transition between states depends on a joint action (σ_0, σ_1) of P_0 and P_1 . In what follows, Σ_0 will be the set of actions of the controller, and Σ_1 will be the set of actions of the environment (or disturbance).

A *system trajectory* is an infinite sequence of states and actions, $(\gamma, s_0, s_1) \in Q^\omega \times \Sigma_0^\omega \times \Sigma_1^\omega$, which satisfies:

$$\gamma[0] \in Q_0 \text{ and } \gamma[i+1] \in \delta(\gamma[i], s_0[i], s_1[i]) \quad (6)$$

We will consider two kinds of trajectory acceptance conditions: $\Omega = (\Box F)$ (meaning that $\forall i, \gamma[i] \in F$), and its dual $\Omega = (\Diamond G)$ (meaning that $\exists i, \gamma[i] \in G$).

where F and G are subsets of Q . P_0 wins the game if the trajectory satisfies $\Box F$, otherwise P_1 wins. To illustrate the duality between the two kinds of acceptance conditions we assume that P_1 wins the game $\Omega = (\Diamond G)$ if the trajectory satisfies $\Diamond G$.

State Space Partition Consider the acceptance condition $\Omega = (\Box F)$. The *winning states* for P_0 are those states $W^* \subseteq F$ from which P_0 can force the system to stay in F . The set W^* can be calculated as the fixed point of the following iteration (using a negative index $i \in \mathbb{Z}_-$ to indicate that each step is a predecessor operation):

$$\begin{aligned} W^0 &= F \\ W^{i-1} &= W^i \cap \{q \in Q \mid \exists \sigma_0 \in \Sigma_0 \forall \sigma_1 \in \Sigma_1 \delta(q, (\sigma_0, \sigma_1)) \subseteq W^i\} \end{aligned} \quad (7)$$

The iteration terminates when $W^i = W^{i-1} \triangleq W^*$. At each step of the iteration, the set W^i contains those states for which P_0 has a sequence of actions which will ensure that the system remains in F for at least i steps, for all possible actions of P_1 .

Now consider the acceptance condition $\Omega = (\Diamond G)$. The *winning states* for P_1 are those states $V^* \supseteq G$ from which P_1 can force the system to visit G . It can be calculated iteratively by:

$$\begin{aligned} V^0 &= G \\ V^{i-1} &= V^i \cup \{q \in Q \mid \exists \sigma_1 \in \Sigma_1 \forall \sigma_0 \in \Sigma_0 \delta(q, (\sigma_0, \sigma_1)) \subseteq V^i\} \end{aligned} \quad (8)$$

terminating when $V^i = V^{i-1} \triangleq V^*$. Here, V^i contains those states for which P_1 has a sequence of actions which will ensure that the system touches G in at most i steps, for all possible actions of P_0 .

The Value Function For the acceptance condition $\Omega = (\Box F)$, we inductively define a *value function*:

$$J(q, i) : Q \times \mathbb{Z}_- \rightarrow \{0, 1\} \quad (9)$$

by:

$$J(q, i) = \begin{cases} 1 & q \in W^i \\ 0 & q \in (W^i)^c \end{cases} \quad (10)$$

In other words, $W^i = \{q \in Q \mid J(q, i) = 1\}$. Recall that P_0 is trying to keep the system in F while P_1 is trying to force the system to leave F . Therefore,

$$\max_{\sigma_0} \min_{\sigma_1} \min_{q' \in \delta(q, \sigma_0, \sigma_1)} J(q', i) = \begin{cases} 1 & \text{if } \exists \sigma_0 \text{ that } \forall \sigma_1, \delta(q, \sigma_0, \sigma_1) \subseteq W^i \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

The $\min_{q' \in \delta(q, \sigma_0, \sigma_1)}$ in the above compensates for the nondeterminism in δ , and the notation $\max_{\sigma_0} \min_{\sigma_1}$ means that P_0 *plays first*, trying to maximize the

minimum value of $J(\cdot)$. P_1 has the advantage in this case, since it has "prior" knowledge of P_0 's action when making its own choice. Therefore, in general,

$$\max_{\sigma_0} \min_{\sigma_1} \min_{q' \in \delta(q, \sigma_0, \sigma_1)} J(\cdot) \leq \min_{\sigma_1} \max_{\sigma_0} \min_{q' \in \delta(q, \sigma_0, \sigma_1)} J(\cdot) \quad (12)$$

with equality occurring when the action (σ_0, σ_1) is a *saddle solution*, or a *no regret* solution for each player. Here we do not need to assume the existence of a saddle solution, rather we always give advantage to P_1 , the player doing its worst to drive the system out of F .

The iteration process (7) may be summarized by the difference equation:

$$J(q, i-1) - J(q, i) = \min\{0, \max_{\sigma_0} \min_{\sigma_1} [\min_{q' \in \delta(q, \sigma_0, \sigma_1)} J(q', i) - J(q, i)]\} \quad (13)$$

which describes the relationship between the change in $J(\cdot)$ due to one step of the iteration and the change in $J(\cdot)$ due to one state transition. The first "min" in equation (13) prevents states outside W^i that can be forced by P_0 to transition into W^i from appearing in W^{i-1} .

To calculate the set of winning states W^* for P_0 we iterate equation (13) until a fixed point is reached, i.e. until for all $q \in Q$, $J(q, i-1) = J(q, i) \triangleq J^*(q)$.

Proposition 1 (Winning States for P_0) *A fixed point $J^*(q)$ of (13) is reached in a finite number of iterations. The set of winning states for P_0 is $W^* = \{q \in Q | J^*(q) = 1\}$.*

Definition 1 (Σ_0 -controlled invariant set) *A subset $W \subseteq Q$ is called Σ_0 -controlled invariant if $\forall s_1 \in \Sigma_1^\omega, \exists s_0 \in \Sigma_0^\omega$ such that for the system trajectory $(\gamma, s_0, s_1) \in Q^\omega \times \Sigma_0^\omega \times \Sigma_1^\omega$, γ remains in W .*

Proposition 2 (Characterization of W^*) *W^* is the largest Σ_0 -controlled invariant subset of F .*

A feedback controller for σ_0 that renders W^* invariant can now be constructed. For all $q \in W^*$ the controller allows only the $\sigma_0 \in \Sigma_0$ for which:

$$\min_{\sigma_1} \min_{q' \in \delta(q, \sigma_0, \sigma_1)} J^*(q') = 1$$

Existence of such σ_0 for all $q \in W^*$ is guaranteed by construction. This control scheme is in fact "least restrictive".

An algorithm for calculating V^* can be constructed similarly. If $G = F^c$, the second game ($\Omega = (\diamond G)$) is the *dual* of the first game ($\Omega = (\Box F)$) in the sense that if the sequence of actions $(s_0, s_1) \in \Sigma_0^\omega \times \Sigma_1^\omega$ of the first game is a *saddle* or *no regret* solution, then $V^* = (W^*)^c$.

3.2 Dynamic Games on Nonlinear Continuous Systems

Consider now the dynamic counterpart of the above class of discrete games: two-player zero-sum dynamic games on nonlinear continuous-time systems. The acceptance conditions considered here correspond to a class of dynamics games known as *pursuit-evasion* games. Player 0 wins if it can keep the system from entering a "bad" subset of the state space, called the *capture set*. Player 1 wins if it can drive the state into the bad set (if it can capture Player 0). As in the previous section, we describe the calculation of the set of states from which Player 0 can always win.

System Definition and Winning Condition As in the discrete case, we consider two players P_0 and P_1 , but now over nonlinear systems of the form

$$\dot{x}(t) = f(x(t), u(t), d(t)) \quad (14)$$

where $x \in \mathbb{R}^n$ is the finite-dimensional *state space*, $u \in U \subseteq \mathbb{R}^u$ is the *control input* which models the actions of P_0 , $d \in D \subseteq \mathbb{R}^d$ is the *disturbance input* which models the actions of P_1 , and f is a smooth vector field over \mathbb{R}^n . The input set $U \times D$ is the analog of the partition $\Sigma_0 \times \Sigma_1$ of the discrete game. The space of acceptable control and disturbance trajectories are denoted by $\mathcal{U} = \{u(\cdot) \in PC^0 \mid u(\tau) \in U \forall \tau \in \mathbb{R}\}$, $\mathcal{D} = \{d(\cdot) \in PC^0 \mid d(\tau) \in D \forall \tau \in \mathbb{R}\}$.

A *system trajectory* over an interval $I \subseteq \mathbb{R}$ is a map:

$$(x(\cdot), u(\cdot), d(\cdot)) : I \rightarrow \mathbb{R}^n \times U \times D \quad (15)$$

such that $u(\cdot) \in \mathcal{U}$, $d(\cdot) \in \mathcal{D}$, $x(\cdot)$ is continuous and $\forall \tau \in I$ where $u(\cdot)$ and $d(\cdot)$ are continuous, $\dot{x}(\tau) = f(x(\tau), u(\tau), d(\tau))$. We assume that the function f is *globally Lipschitz* in x and continuous in u and d . Then, by the existence and uniqueness theorem of solutions for ordinary differential equations, given an interval I , the value of $x(t)$ for some $t \in I$ and input and disturbance trajectories $u(\tau), d(\tau)$ over I there exists a unique solution $x(\cdot, u(\cdot), d(\cdot))$ to (14).

We define the capture set as a region G by $G = \{x \in \mathbb{R}^n \mid l(x) < 0\}$ with boundary $\partial G = \{x \in \mathbb{R}^n \mid l(x) = 0\}$ where $l : \mathbb{R}^n \rightarrow \mathbb{R}$ is a differentiable function of x and $Dl(x) \neq 0$ on ∂G . Defining $F = G^c$, we say P_0 wins the game if for all $\tau \in \mathbb{R}$, $x(\tau) \in F$.

State Space Partition The winning states for P_0 are those states $W^* \subseteq \mathbb{R}^n$ from which P_0 can force the system to stay in $F = G^c$. Define the outward pointing normal to G as:

$$\nu = Dl(x) \quad (16)$$

The states on ∂G which can be forced into G infinitesimally constitute the *usable part* (UP) of ∂G [14]. They are the states for which the disturbance can force the vector field to point inside G :

$$\text{UP} = \{x \in \partial G \mid \forall u \exists d \quad \nu^T f(x, u, d) < 0\} \quad (17)$$

Figure 3 displays a simple example, with the UP of ∂G shown in bold.

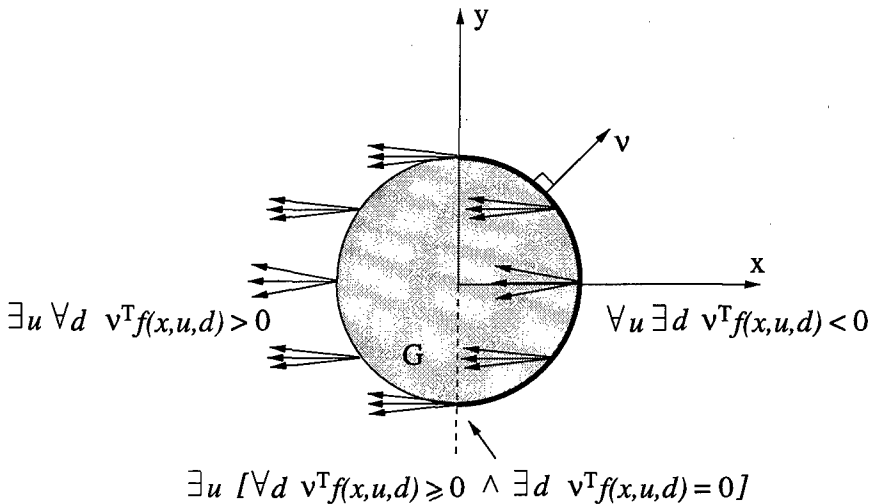


Fig. 3. The capture set G , its outward pointing normal ν , and the cones of vector field directions at points on ∂G .

The Value Function and Hamilton-Jacobi-Bellman equation Consider the system (14) over the time interval $[t, 0]$, where $t < 0$. The value function of the game is defined by:

$$J(x, u(\cdot), d(\cdot), t) : \mathbb{R}^n \times \mathcal{U} \times \mathcal{D} \times \mathbb{R} \rightarrow \mathbb{R} \quad (18)$$

such that $J(x, u(\cdot), d(\cdot), t) = l(x(0))$. This value function may be interpreted as the cost of a trajectory $x(\cdot)$ which starts at x at time $t \leq 0$, evolves according to (14) with input $(u(\cdot), d(\cdot))$, and ends at the final state $x(0)$. Note that the value function depends only on the final state: there is no running cost, or *Lagrangian*. This encodes the fact that we are only interested in whether or not the system trajectory ends in G and are not concerned with intermediate states. The game is won by P_1 if the terminal state $x(0)$ belongs inside G (i.e. $J < 0$), and is won by P_0 otherwise.

Let:

$$u^* = \arg \max_{u \in \mathcal{U}} \min_{d \in \mathcal{D}} J(x, u(\cdot), d(\cdot), t) \quad (19)$$

$$J^*(x, t) = \max_{u \in \mathcal{U}} \min_{d \in \mathcal{D}} J(x, u(\cdot), d(\cdot), t) \quad (20)$$

Thus, the set $\{x : J^*(x, t) \geq 0\}$ contains the states for which the system will stay in $F = G^c$ for at least $|t|$ seconds, regardless of the disturbance d . The continuous-time analog to (7), the iterative method of calculating the winning states for P_0 is therefore:

$$\begin{aligned} W^0 &= G^c \\ W^t &= \{x | J^*(x, t) \geq 0\} \end{aligned} \quad (21)$$

This "iteration" terminates if there exists a $t^* < 0$ such that for all $t < t^*$, $W^t = W^{t^*}$.

We compute $J^*(x, t)$ using standard results in optimal control theory. First, define the *Hamiltonian* of the system as:

$$H(x, p, u, d) = p^T f(x, u, d) \quad (22)$$

where p is a vector in \mathbb{R}^n called the *costate* and is equal to ν at the boundary of G . The optimal Hamiltonian is given by:

$$H^*(x, p) = \max_{u \in U} \min_{d \in D} H(x, p, u, d) \quad (23)$$

If $J^*(x, t)$ is a smooth function of x and t , then it may be calculated for all x and t using the following partial differential equation, known as the *Hamilton-Jacobi-Bellman equation*:

$$-\frac{\partial J^*(x, t)}{\partial t} = \min\{0, H^*(x, \frac{\partial J^*(x, t)}{\partial x})\} \quad (24)$$

with boundary condition $J^*(x, 0) = l(x)$. The derivation of equation (24) may be found in most textbooks on optimal control, for example, see [17]. We added the "min" to the right hand side of (24) for the same reason as in the discrete case: we want to ensure that only the UP of ∂G is propagated backwards, so that states which are once unsafe cannot become safe. Equation (24) is the continuous analog to equation (13) of the preceding discrete game, and describes the relationship between the time and state evolution of $J^*(x, t)$.

Proposition 3 (Winning States for P_0) *If (21) reaches a fixed point at time t^* , then the set of winning states for P_0 is $W^* = \{x | J^*(x, t^*) \geq 0\}$. Otherwise, $W^* = \{x | J^*(x, -\infty) \geq 0\}$. In both cases, $J^*(x, t)$ is the solution of equation (24).*

Definition 2 (U -controlled invariant set) *A subset $W \subseteq \mathbb{R}^n$ is called U -controlled invariant if $\exists u(\cdot) \in U$ such that $\forall d(\cdot) \in D$, $x(\cdot)$ remains in W for the trajectory $(x(\cdot), u(\cdot), d(\cdot))$.*

Proposition 4 (Characterization of W^*) *W^* is the largest U -controlled invariant set contained in $F = G^c$.*

A feedback controller for u that renders W^* invariant can now be constructed. The controller should be such that on ∂W^* only the u for which:

$$\min_{d \in D} \left(\frac{\partial J^*(x, -\infty)}{\partial x} \right)^T f(x, u, d) \geq 0$$

are applied. In the interior of W^* u is free to take on any value in U . Existence of such u 's for $x \in W^*$ is guaranteed by construction. This scheme is in fact least restrictive.

4 Controller synthesis for nonlinear hybrid systems

Nonlinear Hybrid Automata A *hybrid automaton* is a tuple: $H = ((Q \times X), (U \times D), (\Sigma_0 \times \Sigma_1), f, \delta, Inv, (Q_0 \times X_0), \Omega)$ where Q is a finite set of locations, $X = \mathbb{R}^n$, $U \subseteq \mathbb{R}^u$, $D \subseteq \mathbb{R}^d$, $\Sigma = \Sigma_0 \times \Sigma_1$ a finite set of actions, $f : Q \times X \times U \times D \rightarrow \mathbb{R}^n$, $\delta : Q \times X \times \Sigma_0 \times \Sigma_1 \rightarrow 2^{Q \times X}$, $Inv \subseteq Q \times X$, $Q_0 \times X_0 \subseteq Q \times X$ is a subset of initial states, and Ω is an acceptance condition (here $\Omega = (\Box F)$ or $\Omega = (\Diamond G)$ for $F, G \subseteq Q \times X$).

The variables of the hybrid automaton evolve continuously as well as in discrete jumps. A *hybrid time trajectory*, τ , is a finite or infinite sequence of intervals $\tau = \{I_i\}$ satisfying:

- I_i is closed unless τ is finite and I_i is the last interval in the sequence, in which case I_i can be right open.
- Let $I_i = [\tau_i, \tau'_i]$. Then $\tau_0 = 0$ and for all i , $\tau_i = \tau'_{i-1}$, $\tau_i \leq \tau'_i$.

We denote by \mathcal{T} the set of all hybrid time trajectories.

A *system trajectory* is a collection $(\tau, (\gamma(\cdot), x(\cdot)), (u(\cdot), d(\cdot)), (s_0, s_1))$ where $\tau \in \mathcal{T}$, $\gamma : \tau \rightarrow Q$, $x(\cdot) : \tau \rightarrow X$, $u(\cdot) : \tau \rightarrow U$, $d(\cdot) : \tau \rightarrow D$, $s_0 \in \Sigma_0^\omega$ and $s_1 \in \Sigma_1^\omega$ and:

- **Initial Condition:** $(\gamma(\tau_0), x(\tau_0)) \in Q_0 \times X_0$
- **Discrete Evolution:** for all i , $(\gamma(\tau_{i+1}), x(\tau_{i+1})) \in \delta((\gamma(\tau'_i), x(\tau'_i)), (s_0[i], s_1[i]))$.
- **Continuous Evolution** if $\tau'_i > \tau_i$, then for all $t \in [\tau_i, \tau'_i]$, $\gamma(t) = \gamma(\tau_i)$, $\dot{x}(t) = f((\gamma(t), x(t)), (u(t), d(t)))$ and $(\gamma(t), x(t)) \in Inv$.

To ensure that the laws for continuous evolution are meaningful we impose the same assumption on f as in the previous section.

Calculating the Maximal Safe Set Consider the acceptance condition $\Omega = (\Box F)$. We again seek to construct the largest set of states for which the control (in this case both u and σ_0) can guarantee that the acceptance condition Ω is met despite the action of the disturbance (in this case d and σ_1). For any set $K \subseteq Q \times X$ define the controllable and uncontrollable predecessors of K by:

$$\begin{aligned} Pre_0(K) &= \{(q, x) \in Q \times X \mid \exists \sigma_0 \in \Sigma_0 \forall \sigma_1 \in \Sigma_1 \delta((q, x), (\sigma_0, \sigma_1)) \subseteq K\} \cap K \\ Pre_1(K) &= \{(q, x) \in Q \times X \mid \forall \sigma_0 \in \Sigma_0 \exists \sigma_1 \in \Sigma_1 \delta((q, x), (\sigma_0, \sigma_1)) \cap K^c \neq \emptyset\} \cup K^c \end{aligned} \quad (25)$$

In other words, the controllable predecessor of K , $Pre_0(K)$, contains all states in K for which the controllable actions can force the state to remain in K for at least one step in the discrete evolution. The uncontrollable predecessor, on the other hand, contains all states in K^c as well as all states from which the uncontrollable actions may be able to force the state outside K . Clearly:

Proposition 5 $Pre_0(K) \cap Pre_1(K) = \emptyset$.

Consider the algorithm:

Initialization: $W^0 = F$, $W^{-1} = \emptyset$, $i = 0$.

While $W^i \neq W^{i-1}$ **do**

$$W^{i-1} = W^i \setminus \{(q, x) \in Q \times X \mid \forall u \in \mathcal{U} \exists t \geq 0, d \in \mathcal{D} \text{ such that} \\ (\gamma(t), x(t)) \in \text{Pre}_1(W^i) \text{ and } (\gamma(\tau), x(\tau)) \notin \text{Pre}_0(W^i)\}$$

$$i = i - 1$$

end

Here $(\gamma(\tau), x(\tau))$ for $\tau \in [0, t]$ represents the continuous trajectory starting at (q, x) under inputs (u, d) , i.e. $(\gamma(0), x(0)) = (q, x)$ and for all τ , $\dot{\gamma}(\tau) = q$, $\dot{x}(\tau) = f((\gamma(\tau), x(\tau)), (u(\tau), d(\tau)))$, and $(\gamma(\tau), x(\tau)) \in \text{Inv}$.

The most challenging part of each step of the algorithm is the computation of the set of states that can be driven by d to $\text{Pre}_1(W^i)$ without first entering $\text{Pre}_0(W^i)$. This computation can be carried out by appropriately modifying the Hamilton-Jacobi-Bellman construction of Section 3.2.

5 Computational Issues

In practice, the usefulness of the proposed synthesis algorithm depends on our ability to efficiently compute solutions the Hamilton-Jacobi-Bellman equation. We conclude this paper with a brief discussion of some of the computational issues which we are currently investigating.

Numerical methods for computing solutions to the Hamilton-Jacobi-Bellman PDE have been studied extensively: a survey paper [18] presents a set of computation schemes based on a *level set method* for propagating curves, which uses numerical techniques derived from conservation laws. The approach requires gridding the state space, so while these techniques have been shown to be efficient in two- or three-dimensions, they may become cumbersome in higher dimensions. Also, it is essential that a bound on the error due to approximation be known at each step of the algorithm, in order to guarantee that the computed surface is a conservative approximation to the actual surface.

Numerical solutions are potentially complicated by the fact that the right hand side of equation (24) is non-smooth. This is possibly also the case for the optimal Hamiltonian $H^*(x, p)$. Moreover, as t evolves the solution $J^*(x, t)$ to the Hamilton-Jacobi-Bellman equation can develop discontinuities (known as *shocks*) as a function of x . Finally, it is unreasonable to assume that the capture set is always described by a level set of a single differentiable function $l(x)$: more generally, we should assume that there exists a collection of differentiable functions $l_i(x)$ where $i = 1 \dots m$ such that the capture set is described by $G = \bigcap_{i=1}^m \{x \in \mathbb{R}^n \mid l_i(x) \leq 0\}$. Computing solutions with discontinuous Hamiltonian functions is dealt with in [18] using an evolution function which varies across the grid space. Methods to compute solutions in the presence of shocks are presented in [19], and a "viscosity" method to avoid shocks is presented in [20].

References

1. A. Church, "Logic, arithmetic, and automata," in *Proceedings of the International Congress of Mathematicians*, pp. 23–35, 1962.
2. J. von Neumann and O. Morgenstern, *Theory of games and economic behavior*. Princeton university press, 1947.
3. J. R. Büchi and L. H. Landweber, "Solving sequential conditions by finite-state operators," in *Proceedings of the American Mathematical Society*, pp. 295–311, 1969.
4. M. O. Rabin, "Automata on infinite objects and Church's problem," in *Regional Conference Series in Mathematics*, 1972.
5. A. Puri, *Theory of Hybrid Systems and Discrete Event Systems*. PhD thesis, Department of Electrical Engineering, University of California, Berkeley, California, 1995.
6. W. Thomas, "Automata on infinite objects," in *Formal Models and Semantics, volume B of Handbook of Theoretical Computer Science*, Elsevier Science, 1990.
7. W. Thomas, "On the synthesis of strategies in infinite games," in *Proceedings of STACS 95, Volume 900 of LNCS* (E. W. Mayr and C. Puech, eds.), pp. 1–13, Munich: Springer Verlag, 1995.
8. O. Maler, A. Pnueli, and J. Sifakis, "On the synthesis of discrete controllers for timed systems," in *STACS 95: Theoretical Aspects of Computer Science* (E. W. Mayr and C. Puech, eds.), Lecture Notes in Computer Science 900, pp. 229–242, Munich: Springer Verlag, 1995.
9. E. Asarin, O. Maler, and A. Pnueli, "Symbolic controller synthesis for discrete and timed systems," in *Proceedings of Hybrid Systems II, Volume 999 of LNCS* (P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, eds.), Cambridge: Springer Verlag, 1995.
10. H. Wong-Toi, "The synthesis of controllers for linear hybrid automata," in *Proceedings of the IEEE Conference on Decision and Control*, (San Diego, CA), 1997.
11. W. M. Wonham, *Linear Multivariable Control: a geometric approach*. Springer Verlag, 1979.
12. A. Deshpande and P. Varaiya, "Viable control of hybrid systems," in *Hybrid Systems II* (P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, eds.), Lecture Notes in Computer Science 999, pp. 128–147, Berlin: Springer Verlag, 1995.
13. R. Isaacs, *Differential Games*. John Wiley, 1967.
14. T. Başar and G. J. Olsder, *Dynamic Non-cooperative Game Theory*. Academic Press, second ed., 1995.
15. C. Tomlin, G. Pappas, and S. Sastry, "Conflict resolution for air traffic management: A case study in multi-agent hybrid systems," tech. rep., UCB/ERL M97/33, Electronics Research Laboratory, University of California, Berkeley, 1997. To appear in the IEEE Transactions on Automatic Control.
16. J. Lygeros, D. N. Godbole, and S. Sastry, "A verified hybrid controller for automated vehicles," Tech. Rep. UCB-ITS-PRR-97-9, Institute of Transportation Studies, University of California, Berkeley, 1997. To appear in the IEEE Transactions on Automatic Control, Special Issue on Hybrid Systems, April 1998.
17. A. E. Bryson and Y.-C. Ho, *Applied Optimal Control*. Waltham: Blaisdell Publishing Company, 1969.
18. J. A. Sethian, "Theory, algorithms, and applications of level set methods for propagating interfaces," tech. rep., Center for Pure and Applied Mathematics (PAM-651), University of California, Berkeley, 1995.

19. J. M. Berg, A. Yezzi, and A. R. Tannenbaum, "Phase transitions, curve evolution, and the control of semiconductor manufacturing processes," in *Proceedings of the IEEE Conference on Decision and Control*, (Kobe), pp. 3376-3381, 1996.
20. P. L. Lions, *Generalized Solutions of Hamilton-Jacobi Equations*. London: Pittman, 1982.

A Sufficient Condition for Controllability of a Class of Hybrid Systems

Jan H. van Schuppen

CWI

P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

J.H.van.Schuppen@cwi.nl

also affiliated with

Department of Mathematics and Computing Science
Eindhoven University of Technology

Control of engineering systems by computers is formulated as a control synthesis problem for hybrid control systems. An input-output hybrid control system is used as a control-theoretic model for this problem. A sufficient condition for controllability of a hybrid system is formulated that separates into a sufficient condition at the discrete and at the continuous level of a hybrid system.

Keywords and Phrases: Hybrid system, control, discrete-event system, controllability.

1 Introduction

The purpose of this paper is to present results on control synthesis for a particular class of hybrid control systems.

In this paper attention is focused on control problems for hybrid systems in which the discrete-events are subject to control and in which there are no discrete-events generated by the environment. Control problems of this type arise in control of mechanical systems, say robots, and in control of chemical plants. The main characteristic of such problems is that the control is supplied by discrete-input-events and by the continuous input to the system. It differs from control problems in which the environment of the system exclusively supplies the events.

A hybrid control system is a control theoretic model for a computer controlled engineering system. For the purposes of this paper the definition of a hybrid control system in [13] is restricted to an input-output hybrid control system stated below. The definition is inspired by those of [1, 2]. The general control problem for hybrid systems leads to a problem of controllability of such systems. Because of its generality this problem is unlikely to be solvable analytically. Therefore a sufficient condition for controllability is formulated and proven to achieve sufficiency. The sufficient condition separates controllability at the discrete-event level and at the level of the continuous systems. An advantage of this approach is that it is comparatively easier to check the sufficient condition than it is to check general controllability. For controllability of the control systems at the

continuous level results are available, although for nonlinear systems such conditions may be difficult to verify. Controllability conditions for an input-output automaton are in principle straightforward to check although the complexity of this problem may be high. The sufficient condition for controllability of a hybrid control system is not necessary in general. Experience with examples will have to establish whether or not the envisioned advantages are useful in control problems.

Control synthesis for hybrid control systems is discussed in several papers and theses. The publications most closely related to this paper are briefly mentioned. M. Branicky has proposed to use optimal control theory for control synthesis, see [2, 3, 4]. The existence of a controller then follows from the existence of a solution to a set of Bellman-Hamilton-Jacobi equations. Conditions for the existence are not discussed and if formulated may be restrictive or difficult to check. In contrast with that paper, the approach of this paper has explicit conditions for the existence of an input sequence. A. Deshpande and P. Varaiya have a theory on viable control of hybrid control systems, see [5, 6]. The approach of those papers is close to the approach of this paper, the latter paper presenting more explicit controllability conditions. Other references on controller synthesis are [7, 8, 9, 10]. For concepts and results on control theory the reader is referred to the book [12] and on automata and computation referred to the book [11].

A summary of the paper follows. Section 2 contains a definition of an input-output hybrid control system and the problem formulation. Controllability is treated in Section 3. Concluding remarks are stated in Section 4.

2 Problem Formulation and Preliminaries

Remarks on notation follow. Denote the set of the integers by $Z = \{1, 2, \dots\}$ and the natural numbers by $N = \{0, 1, 2, \dots\}$. For $n \in Z$ denote $Z_n = \{1, 2, \dots, n\}$. Denote the set of the real numbers by R and the positive real numbers by $R_+ = [0, \infty)$.

A *continuous-time hybrid control system* is a tuple

$$\left\{ T, Q, \Sigma_{in}, \Sigma_{int}, \Sigma_{cd}, \Sigma_{out}, U, Y, U_c, U_{ex}, \delta, r, \{X_q, TX_q, G_q, f_q, h_q, \forall q \in Q\}, (q_0, x_{q_0,0}) \right\}, \quad (1)$$

where

$T = R_+$, said to be the *time index set*,

Q is a finite set, the *discrete state set*,

Σ_{in} is a finite set, the *set of input events*,

Σ_{int} is a finite set, the *set of internal events*,

Σ_{cd} is a finite set, the *set of events generated by the continuous dynamics*,

$\Sigma = \Sigma_{in} \cup \Sigma_{int} \cup \Sigma_{cd}$,

$U \subseteq R^m$, the *continuous input space*,

$Y \subseteq R^p$ the *continuous output space*,

$U_c \subset \{u : T \rightarrow U\}$, *set of continuous input functions*,

$U_{ex} \subset (T \times \Sigma)^* \cup (T \times \Sigma)^\omega$ the set of external timed-event sequences,
 $\delta: T \times Q \times X \times \Sigma \rightarrow Q$, the discrete *transition function*,
 a , possibly partial, function,
 $r: T \times Q \times X \times \Sigma \rightarrow X$, the *reset map*, a , possibly partial, function,
 for all $q \in Q$,
 $X_q \subseteq R^{n_q}$, the *continuous state space at discrete state* $q \in Q$, $X = \cup_{q \in Q} X_q$,
 $TX_q(x) \subseteq R^{n_q}$ the tangent space at $x \in X_q$,
 $G_q: \Sigma_{cd} \rightarrow P_{closed}(X_q)$, the *guard* at $q \in Q$, a , possibly partial, function,
 $P_{closed}(X_q)$ denotes the closed subsets of X_q ,
 $f_q: T \times X_q \times U \rightarrow TX_q$, $h_q: T \times X_q \times U \rightarrow Y$,
 are functions that determine a differential equation and a read-out map,
 $(q_0, x_{q_0,0}) \in Q \times X_{q_0}$ the *initial state*.

The dynamics of the hybrid control system is described by the discrete transition function, the reset map, the differential equation, and the output map, according to

$$q^+ = \delta(t, q^-, x_{q^-}^-, \sigma), \quad q_0, \quad (2)$$

$$x_{q^+}^+ = r(t, q^-, q^+, x_{q^-}^-, \sigma), \quad (3)$$

$$\dot{x}_q(t) = f_q(t, x_q(t), u(t)), \quad x_q(0) = x_q^+, \quad (4)$$

$$y(t) = h_q(t, x_q(t), u(t)). \quad (5)$$

The operation of the hybrid control system is described below. At $t = 0$ the initial state is $(q_0, x_{q_0,0}) \in Q \times X_{q_0}$. Assume no immediate transition takes place at $t = 0$. At the discrete state $q = q_0$ the continuous dynamics proceeds according to the differential equation (4). It will be assumed that for all $u \in U_c$ this differential equation has an unique solution on R_+ . The solution will be followed till the next event. The time interval till the next event will be denoted by $[t_0, t_1)$ for $t_1 \in R_+$ and for subsequent intervals by $[t_n, t_{n+1})$ for $n \in Z_+$.

At any time $t \in T$ an event may occur that results in a change of the discrete state. The possible events at discrete state $q \in Q$ and at time $t \in T$ are:

- an *input event* $\sigma \in \Sigma_{in}$ occurs if such an event is supplied on the input channel;
- an *event generated by the continuous dynamics* $\sigma \in \Sigma_{cd}$ occurs immediately when $x_q(t-) \in G_q(\sigma)$, thus if the state of the system hits a guard. (Here the notation $x_q(t-) = \lim_{s \uparrow t} x_q(s)$ is used.)

If the timed-event (t, σ_1) occurs then the transition is described by the discrete transition function and the reset map (2,3). It may be the case that the new state $(q^+, x_{q^+}^+) \in Q \times X_{q^+}$ is such that $x_{q^+}^+ \in G_{q^+}(\sigma_2)$. In this case the event $\sigma_2 \in \Sigma_{cd}$ takes place at the same time. It will be assumed that only a finite number of events can occur at any time (non-Zenoness). After the last event of the sequence of events occurring at moment t , the new state is $(q_f, x_{q_f}^+)$ where $x_{q_f}^+$ is the initial condition of the differential equation in the discrete state q_f .

A hybrid control system is said to be *time-invariant* if the functions δ, r, f_q, h_q do not depend explicitly on the time index set. In this paper attention is restricted to time-invariant hybrid control systems.

In this paper attention is focused on a control problem for hybrid control systems, as in path planning for robotics. This problem leads to controllability conditions for hybrid systems that may be useful for other control problems.

Problem 1. Consider a time-invariant hybrid control system. For any initial and terminal state, do there exist a timed-event sequence and a sequence of input trajectories

$$\{(t_i, \sigma_i) \in T \times \Sigma_{in}, i = 1, \dots, m_s\}, \quad \{u_i : [t_i, t_{i+1}) \rightarrow U, i = 1, \dots, m_s\},$$

such that, when the system is supplied with these inputs, the system is transferred from the pre-specified initial state to the terminal state?

The input trajectories should preferably be generated by a control law in the form of a controller. The controller itself should also be hybrid, it may be taken to be a hybrid control system. In this paper attention is restricted to the existence of input sequences, not on the construction of the controller.

3 A Sufficient Condition for Controllability

Definition 2. A hybrid control system is said to be *controllable* if for any pair of states $(q_0, x_{q_0,0}), (q_f, x_{q_f,f}) \in Q \times X$ there exists a timed-event sequence and a continuous-input sequence such that the system evolves from the initial state $(q_0, x_{q_0,0})$ to the final state $(q_f, x_{q_f,f})$.

In control theory, controllability of a control system is a sufficient condition for the existence of a controller.

What are necessary and sufficient conditions for a hybrid control system to be controllable? First a general sufficiency condition is presented in terms of arrival sets. Subsequently it is shown how to construct these sets.

Proposition 3. Consider a hybrid control system. Assume there exists a collection of sets, called arrival sets,

$$AR(q) = \{AR(q, i) \subset X_q, \forall i \in Z_{n_q}\}, \quad AR = \{AR(q), \forall q \in Q\},$$

such that

1. for any $(q, x_{q,0}) \in Q \times X$ there exists a continuous input $u \in U_c$ on the interval $[t_0, t_1)$, a timed-event (t_1, σ_1) , possibly further timed-events at t_1 , and $(q_1, AR(q_1, i))$, such that the system is transferred from state $(q, x_{q,0})$ to a state $(q_1, x_{q_1}) \in Q \times AR(q_1, i)$;
2. for any $(q_0, x_{q_0}) \in Q \times AR(q_0, i)$, $q_f \in Q$, and $AR(q_f, j) \in AR(q_f)$ there exists a finite sequence of timed-events and a finite sequence of continuous input signals

$$\{(t_i, \sigma_i) \in T \times \Sigma_{in}, i = 1, \dots, m_s\}, \quad \{u_i : [t_i, t_{i+1}) \rightarrow U, i = 1, \dots, m_s\},$$

- such that the system is transferred from state $(q, x_{q,0})$ to a state $(q_f, x_{q_f}) \in Q \times AR(q_f, j)$;
3. for any $(q_f, x_{q_f,f}) \in Q \times X_{q_f}$ there exists an arrival set $AR(q_f, j)$ and for any state $x_{q_f,0} \in AR(q_f, j)$, a continuous input $u \in U_c$ such that the system is transferred from state $(q_f, x_{q_f,0}) \in Q \times AR(q_f, j)$ to state $(q_f, x_{q_f,f})$ without leaving the state space X_{q_f} .

Then the hybrid control system is controllable.

Proof Consider $(q_0, x_{q_0,0}) \in Q \times X$ and $(q_f, x_{q_f,f}) \in Q \times X$. By condition 1 there exists a continuous input, a timed-event, and a pair $(q_1, AR(q_1, i))$, such that the system is transferred from state (q_0, x_{q_0}) to a state $(q_1, x_{q_1,1}) \in Q \times AR(q_1, i)$. From Condition 3 then follows that there exists a $AR(q_f, j)$ and for any state $x_{q_f,0} \in AR(q_f, j)$ a continuous input such that the system is transferred from state $(q_f, x_{q_f,0}) \in Q \times AR(q_f, j)$ to the state $(q_f, x_{q_f,f})$. From Condition 2 then follows that there exists a finite timed-event sequence and a finite sequence of continuous-input signals such that the system is transferred from any state $(q_1, x_{q_1,1}) \in Q \times AR(q_1, i)$ to a state $(q_f, x_{q_f,0}) \in Q \times AR(q_f, m)$. The three conditions together imply that the system can be transferred from the initial state to the terminal state. \square

The construction of the arrival sets requires the introduction of a few definitions.

Definition 4. Consider a time-invariant hybrid control system. The *continuous-controllability set* at state $(q, x_{q,1}) \in Q \times X$ is defined to be the set

$$C - Con_q(\{x_{q,1}\}) = \left\{ (q, x_{q,0}) \in Q \times X_q \mid \text{either } x_{q,0} = x_{q,1} \right. \\ \left. \text{or } \exists t_0, t_1 \in R_+, t_0 < t_1, u \in U_c, \text{ such that } \right. \\ \left. x_q(t_0) = x_{q,0}, x_q(t_1) = x_{q,1}, \text{ and } \forall t \in [t_0, t_1), x_q(t) \in X_q \right\}.$$

In words, $C - Con_q(\{x_{q,1}\})$ is the subset of the state space $\{q\} \times X_q$ from which one can start and by application of a continuous input arrive at the state $(q, x_{q,1})$ without ever leaving the state space X_q in the interval $[t_0, t_1)$. Note that the definition explicitly excludes the possibility that the state trajectory hits a guard because if it did so then the system would move to another discrete state and thus leave the state space X_q .

Let for $S_q \subseteq X_q$ the *continuous-controllability set* of S_q be defined by

$$C - Con_q(S_q) = \cup_{x_{q,1} \in S_q} C - Con_q(\{x_{q,1}\}).$$

Then for $\sigma \in \Sigma_{cd}$

$$C - Con_q(G_q(\sigma)) = \left\{ (q, x_{q,0}) \in Q \times X \mid \text{either } x_{q,0} \in G_q(\sigma) \right. \\ \left. \text{or } \exists t_0, t_1 \in R_+, t_0 < t_1, u \in U_c, \text{ such that } \right. \\ \left. x_q(t_0) = x_{q,0}, x_q(t_1-) \in G_q(\sigma), \text{ and } \forall t \in [t_0, t_1), x_q(t) \in X_q \right\}.$$

Definition 5. Consider a time-invariant hybrid control system.

- a. For $q_i, q_j \in Q$ and $\sigma \in \Sigma_{in}$ define the *departure set* $D(q_i, \sigma, q_j, A_{q_j}^+)$ and the collection of departure sets $D(q_i)$ as

$$D(q_i, \sigma, q_j, A_{q_j}^+) = \left\{ x_{q_i} \in X_{q_i} \mid \exists x_{q_j} \in A_{q_j}^+ \text{ such that } \begin{aligned} & q_j = \delta(q_i, x_{q_i}, \sigma) \text{ and } x_{q_j} = r(q_i, q_j, x_{q_i}, \sigma) \end{aligned} \right\},$$

$$D(q_i) = \left\{ D(q_i, \sigma, q_j, A_{q_j}^+), \forall q_j \in Q, \forall \sigma \in \Sigma_{in}, \forall A_{q_j}^+ \subseteq X_{q_j} \right\}.$$

In words, $D(q_i, \sigma, q_j, A_{q_j}^+)$ consists of those states in X_{q_i} at which, when the input event occurs at the time the continuous state is in this subset of the state space, the system is transferred to the discrete state q_j and to a continuous state in $A_{q_j}^+$. Note that in the definition of $D(q_i, \sigma, q_j, A_{q_j}^+)$ both q_j and σ are required. Because an input event can occur at any time it follows that for $q_i \in Q$ and $\sigma \in \Sigma_{in}$ the set

$$\{D(q_i, \sigma, q_j, X_{q_j}) \subset X_{q_i}, q_j \in Q\},$$

forms a partition of the state space X_{q_i} . Let for $q_i, q_j \in Q$ and $\sigma \in \Sigma_{in}$

$$C - Con_{q_i}(D(q_i, \sigma, q_j, X_{q_j}))$$

be the continuous-controllability set of $D(q_i, \sigma, q_j, X_{q_j})$. In words, this set consists of all states in X_{q_i} from which one can transfer the system by a continuous input to a state in $D(q_i, \sigma, q_j, X_{q_j})$ at which the input event $\sigma \in \Sigma_{in}$ can be applied and as a consequence the system moves to the new discrete state q_j .

- b. For $q_i, q_j \in Q$ and $\sigma \in \Sigma_{in} \cup \Sigma_{cd}$ define the *arrival set* as

$$AR(q_i, \sigma, q_j) = \left\{ x_{q_j}^+ \in X_{q_j} \mid \exists x_{q_i}^- \in X_{q_i} \text{ such that } \begin{aligned} & q_j = \delta(q_i, x_{q_i}^-, \sigma) \text{ and } x_{q_j}^+ = r(q_i, q_j, x_{q_i}^-, \sigma) \end{aligned} \right\}.$$

In words, $AR(q_i, \sigma, q_j)$ consists of those states in X_{q_j} at which one arrives in set X_{q_j} from the discrete state q_i by an event $\sigma \in \Sigma_{in} \cup \Sigma_{cd}$. Denote

$$AR = \{AR(q_i, \sigma, q_j) \subseteq X_{q_j}, \forall q_i, q_j \in Q, \sigma \in \Sigma_{in} \cup \Sigma_{cd}\}.$$

Terminology of discrete-event systems is introduced. A discrete-event system is defined to be a generator consisting of the objects (Q, Σ, δ, q_0) , where Q is a finite set called the state set, Σ is a finite set called the alphabet, $\delta : Q \times \Sigma \rightarrow Q$ is a function called the transition function (it may be a partial function), and $q_0 \in Q$ is the initial state of the system. Denote by Σ^* the set of all finite strings with events in Σ and the empty string $\epsilon \notin \Sigma$. Extend the transition function to $\delta : Q \times \Sigma^* \rightarrow Q$ by defining it for sequences.

The discrete-event system is said to be *reachable* if for any $q_1 \in Q$ there exists a $s \in \Sigma^*$ such that $q_1 = \delta(q_0, s)$.

Definition 6. Consider a time-invariant hybrid control system. Define the associated *arrival discrete-event system* (arrival DES; actually, it is a generator)

as

$$\begin{aligned}
& (AR, \Sigma_{in} \cup \Sigma_{cd}, \delta_{AR}, AR_{q_0}), \\
& AR = \{AR(q_i, \sigma, q_j) \subseteq X_{q_j}, \forall q_i, q_j \in Q, \sigma \in \Sigma_{in} \cup \Sigma_{cd}\}, \\
& AR(q_i, \sigma_1, q_j) = \delta_{AR}(AR(q_k, \sigma_0, q_i), \sigma_1) \\
& \text{if } \begin{cases} \text{either } AR(q_k, \sigma_0, q_i) \subseteq C - Con_{q_i}(G_{q_i}(\sigma_1)) \\ \text{or } AR(q_k, \sigma_0, q_i) \subseteq C - Con_{q_i}(D(q_i, \sigma_1, q_j, X_{q_j})), \end{cases} \\
& \text{else not defined,} \\
& AR_{q_0} = AR(q_j, \sigma, q_0) \text{ if } x_{q_0,0} \in AR(q_j, \sigma, q_0) \in AR.
\end{aligned}$$

In words, the transition takes place if the arrival set $AR(q_k, \sigma_0, q_i)$ is fully contained in one of the indicated continuous-controllability sets. In this case it is possible to start in any state of the arrival set and to transfer the system to either a guard or to a departure set at which state an event occurs or can be supplied to the system respectively.

Note that the conditions in the definitions of the transition function δ_{AR} are restrictions, it may be the case that $AR(q_k, \sigma_0, q_i)$ is not fully contained in one set but intersects with two or more continuous-controllability sets. Thus, a particular arrival set may not be related by a transition to any of the other arrival sets in the arrival DES.

Proposition 7. *Consider a time-invariant hybrid control system. If*

1. *the associated arrival DES $(AR, \Sigma_{in} \cup \Sigma_{cd}, \delta_{AR}, AR_{q_0})$ is reachable;*
- 2.

$$\begin{aligned}
& \forall (q_f, x_{q_f,f}) \in Q \times X, \exists q_i \in Q, \sigma \in \Sigma_{in} \cup \Sigma_{cd} \text{ such that} \\
& AR(q_i, \sigma, q_f) \subseteq C - Con_{q_f}(\{x_{q_f,f}\});
\end{aligned}$$

in words, any final state can be reached from any state in an arrival set in X_{q_f} associated with either an input event or an event generated by the continuous dynamics;

then the hybrid control system is controllable.

Proof This follows from Proposition 3 and the Definitions 4, 5, and 6. \square

The main difficulty in the application of the above result is to determine the controllability sets. The control system at a particular discrete state will in general be nonlinear and may have a geometrically structured input space. In specific cases the controllability set can be approximated. Note that because Proposition 7 only describes a sufficient condition for controllability it is possible to take smaller subsets than $C - Con_q(G_q(\sigma))$ and $C - Con_q(D(q_i, \sigma, q_j, X_{q_j}))$ in the definition of the AR system.

As remarked above, the condition imposed in Definition 6 is restrictive because the complete arrival set has to be contained in either the controllability set of a

guard or in the destination set of an input event. Below a different approach is described.

Consider the finite collection

$$A = \{A(q_i, k) \subset X_{q_i}, k = 1, \dots, n_{q_i}, q_i \in Q\}.$$

Assume that there exists a $A_0 = A(q_0, r) \in A$ such that $x_{q_0,0} \in A(q_0, r)$. Define for a hybrid control system, $q_i, q_j \in Q$, $\sigma \in \Sigma_{in} \cup \Sigma_{cd}$, and $A \subset X_{q_j}$,

$$DT(q_i, \sigma, q_j, A) = \left\{ \begin{array}{l} x_{q_i} \in X_{q_i} | \\ q_j = \delta(q_i, x_{q_i}, \sigma) \text{ and } r(q_i, q_j, x_{q_i}, \sigma) \in A \end{array} \right\}.$$

The definition of the set DT is a minor extension of that of D . Note that for $\sigma \in \Sigma_{cd}$,

$$G_{q_i}(\sigma) \subset \cup_{q_j \in Q} DT(q_i, \sigma, q_j, X_{q_j}).$$

Define the generator

$$\begin{aligned} &(A, \Sigma_{in} \cup \Sigma_{cd}, \delta_A, A_0), \\ &A(q_j, m) = \delta_A(A(q_i, k), \sigma), \\ &\text{if } A(q_i, k) \subseteq C - \text{Con}_{q_i}(DT(q_i, \sigma, q_j, A(q_j, m))). \end{aligned}$$

The generator is meaningful because the subset inclusion allows for any state in the set $A(q_i, k)$ the existence of a continuous input that moves the state of the system to the set DT from which an event will transfer the system to a state in the set $A(q_j, m)$.

Theorem 8. Consider a time-invariant hybrid control system. If there exists a collection of sets

$$A = \{A(q_i, k) \subset X_{q_i}, k \in Z_{n_{q_i}}, q_i \in Q\}.$$

such that

1. for all $q_i, q_j \in Q$ and $\sigma \in \Sigma_{in} \cup \Sigma_{cd}$ there holds

$$AR(q_i, \sigma, q_j) = \cup_{k \in I(q_i, \sigma, q_j)} A(q_i, k),$$

for an index set $I(q_i, \sigma, q_j) \subset Z_{n_{q_i}}$;

2. the generator $(A, (\Sigma_{in} \cup \Sigma_{cd}), \delta_A, A_0)$ is reachable;
3. for any $(q_f, x_{q_f, f}) \in Q \times X_{q_f}$ there exists a set $A(q_f, i) \in A$ and for any state $(q_f, x_{q_f, 0}) \in A(q_f, i)$ a continuous input $u \in U_c$ such that the system is transferred from state $(q_f, x_{q_f, 0}) \in Q \times A(q_f, i)$ to state $(q_f, x_{q_f, f})$ without leaving the state space X_{q_f} ;

then the hybrid control system is controllable.

The proof follows along the lines of that of Proposition 3.

A dynamic programming-like procedure can be formulated for the construction of the collection of the A sets. Suppose specified a terminal state $(q_f, x_{q_f}) \in Q \times X_{q_f}$. Construct successively by a backward recursion the collections of sets A_0, A_1, A_2, \dots as follows. Let

$$A_0(q_f, r(q_k, \sigma)) = AR(q_k, \sigma, q_i) \cap C - Con_{q_f}(\{x_{q_f}\}),$$

$$\forall q_k \in Q, \sigma \in \Sigma_{in} \cup \Sigma_{cd},$$

$$A_0(q_i, r(q_k, \sigma)) = AR(q_k, \sigma, q_i), \text{ if } q_i \neq q_f, \forall q_k \in Q, \sigma \in \Sigma_{in} \cup \Sigma_{cd}.$$

For $k = 0, 1, \dots, q_i, q_j \in Q, r_1, r_2 \in Z_+, \sigma \in \Sigma_{in} \cup \Sigma_{cd}$, let

$$A_{k+1}(q_i, (r_1, r_2, s, \sigma_1, \sigma_2, q_j))$$

$$= A_k(q_i, r_1) \cap AR(q_s, \sigma_1, q_i) \cap C - Con_{q_i}(DT(q_i, \sigma_2, q_j, A_k(q_j, r_2))).$$

The range of r_1 is the index set for which $A_k(q_i, \cdot)$ is defined and similarly the range of r_2 is associated with $A_k(q_j, \cdot)$. After the sets $A_{k+1}(q_i, (\dots))$ have all been determined they should be relabeled $r = 1, 2, \dots, n_r$. In general there is no condition that implies that the procedure will terminate after a finite number of steps. However, if it terminates then it still has to be checked whether the conditions of Theorem 8 hold. The procedure formulated above is analogous to but different from a procedure formulated in [5].

The sufficient conditions for controllability were developed with a particular hybrid system in mind. The example has been omitted from the paper because it requires a large amount of space for tables and notation. The reader is referred to the report [13] for a hybrid system of a model of conveyor belts and to [7] for a hybrid system of a model of a chemical plant.

4 Conclusion

Several sufficient conditions for controllability of hybrid control systems have been formulated and proven. Further research is required to test the usefulness of the conditions on examples.

References

1. R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.
2. M.S. Branicky. *Studies in hybrid systems: Modeling, analysis, and control*. PhD thesis, M.I.T., Cambridge, MA, 1995.
3. M.S. Branicky. On-line, reflexive constraint satisfaction for hybrid systems: First Steps. In O. Maler, editor, *Hybrid and real-time systems - Proceedings of International Workshop HART97*, number 1201 in Lecture Notes in Computer Science, pages 93–107, Berlin, 1997. Springer.

4. M.S. Branicky, V.S. Borkar, and S.K. Mitter. A unified framework for hybrid control: Background, model, and theory. Report LIDS-P-2239, Laboratory for Information and Decision Systems, M.I.T., Cambridge, MA, 1994.
5. A. Deshpande and P. Varaiya. Information structures for control and verification of hybrid systems. In *Proceedings American Control Conference*, pages 2642–2647. American Control Council, 1995.
6. A. Deshpande and P. Varaiya. Viable control of hybrid systems. In P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors, *Hybrid systems II*, volume 999 of *Lecture Notes in Computer Science*, pages 128–147, Berlin, 1995. Springer.
7. J.J.H. Fey. Control and verification of industrial hybrid systems using models specified with the formalism χ . Note BS-N9601, CWI, Amsterdam, 1996.
8. M. Heymann, Feng Lin, and G. Meyer. Control synthesis for a class of hybrid systems subject to configuration-based safety constraints. In O. Maler, editor, *Hybrid and real-time systems - Proceedings of International Workshop HART97*, number 1201 in *Lecture Notes in Computer Science*, pages 376–390, Berlin, 1997. Springer.
9. J. Lygeros. *Hierarchical, hybrid control of large scale systems*. PhD thesis, University of California, Berkeley, 1996.
10. J. Lygeros, C. Tomlin, and S. Sastry. Multiobjective hybrid controller synthesis. In O. Maler, editor, *Hybrid and real-time systems - Proceedings of International Workshop HART97*, number 1201 in *Lecture Notes in Computer Science*, pages 109–123, Berlin, 1997. Springer.
11. M. Sipser. *Introduction to the theory of computation*. PWS Publishing Company, Boston, 1997.
12. E.D. Sontag. *Mathematical control theory: Deterministic finite dimensional systems*. Springer-Verlag, New York, 1990.
13. J.H. van Schuppen. Control for a class of hybrid systems. Report PNA-R9716, CWI, Amsterdam, 1997.

Hybrid Regular Expressions*

Li Xuandong, Zheng Tao, Hou Jianmin, Zhao Jianhua, and Zheng Guoliang

State Key Laboratory of Novel Software Technology
Department of Computer Science and Technology
Nanjing University, Nanjing
Jiangsu, P.R.China 210093
E-Mail: lxd@nju.edu.cn

Abstract. In this paper, we consider the problem verifying hybrid systems modelled by linear hybrid automata. We extend the traditional regular expressions with time constraints and use them as a language to describe the behaviour of a class of linear hybrid automata. The extended notation is called *Hybrid Regular Expression* (HRE). Based on linear programming, we show that for the class of linear hybrid automata whose behaviour can be represented by HREs, two class of reachability problems and the satisfaction problem for linear duration invariants are decidable.

1 Introduction

The formalism of hybrid automata [1] have become a standard model for real-time and hybrid systems. A class of hybrid systems can be modelled by linear hybrid automata. Informally, a linear hybrid automaton is a conventional automaton extended with a set of variables, which are used to model the state of the continuous component of hybrid systems and are assumed to be piecewise linear functions of time. The states of the automaton called *locations* are assigned with a change rate for each variable, such as $\dot{x} = w$ (x is a variable, w is a real number), and the transitions of the automaton are labelled with constraints on the variables such as $a \leq x \leq b$ and /or with reset actions such as $x := c$ (x is a variable, a , b , and c are real numbers). Each location is also assigned with an invariant condition that must hold when the system resides at the location. The automaton starts at one of the initial locations with all variables initialised to their initial values. As time progresses, the values of all variables change continuously according to the rate associated with the current location. At any time, the system can change its current location from s to s' provided that there is a transition ρ from s to s' whose labelling conditions are satisfied by the current value of the variables. With a location change by a transition ρ , all the variables are reset to the new value accordingly by the reset actions labelled on ρ . Transitions are assumed to be instantaneous.

* This work is supported by the National Natural Science Foundation of China and International Institute for Software Technology, The United Nations University (UNU/IIST).

Let us consider an example of a water-level monitor in [2]. The water level in a tank is controlled through a monitor, which continuously senses the water level and turns a pump on and off. The water level changes as a piecewise-linear function of time. When the pump is off, the water level falls by two inches per second; when the pump is on, the water level rises by one inch per second. Suppose that initially the water level is one inch and the pump is on. There is a delay of two seconds from the time that the monitor signals to change the status of the pump to the time that the change becomes effective. The system is modelled by the hybrid automaton depicted in Figure 1. The automaton has four locations. In the locations s_1 and s_2 , the pump is on; in the locations s_3 and s_4 , the pump is off. The variable y is used to model the water-level, and x is used to specify the delays: whenever the control is in location s_2 or s_3 , the value of x indicates how long the signal to switch the pump off or on has been sent.

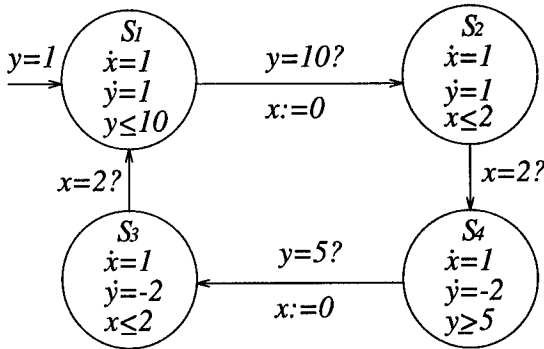


Fig. 1. A hybrid automaton modelling a water-level monitor

In this paper, we use timed sequences to express the behaviour of real-time and hybrid systems. A timed sequence $(s_1, t_1) \wedge (s_2, t_2) \wedge \dots \wedge (s_m, t_m)$ represents a behaviour of a system that the system starts at the state s_1 , stays there for t_1 time units, then changes to s_2 and stays in s_2 for t_2 time units, and so on. The values t_1, t_2, \dots, t_m have to satisfy some time constraints enforced by the system. For example, $(s_1, 9) \wedge (s_2, 2) \wedge (s_4, 3.5) \wedge (s_3, 2) \wedge (s_1, 8)$ expresses a behaviour of the hybrid automaton in Fig. 1.

Since the number of timed sequences to express the behaviour of a system may be infinite, we have to find a notion as a finite representation of behaviour of systems. A traditional way to express the behaviour of an automaton is to use regular expressions. In this paper, we extend the traditional regular expressions with time constraints and use them as a language to describe the behaviour of real-time and hybrid systems. The extended notation will be called *Hybrid Regular Expression* (HRE). HREs can express the behaviour of a class of linear hybrid automata. We show that for the class of linear hybrid automata whose behaviour can be expressed by HREs, two class of reachability problems and the

satisfaction problem for linear duration invariants are decidable.

The paper is organised as follows. In the next section, we introduce the notion of Hybrid Regular Expression. Section 3 shows that two class of reachability problems for the class of linear hybrid automata whose behaviour can be expressed by HREs are decidable. The satisfaction problem of linear duration invariants for this class of linear hybrid automata is resolved in Section 4. The last section is the conclusion of the paper.

2 Hybrid Regular Expressions

While a regular expression over a set of states (alphabet) is a finite representation of a (infinite) set of sequences of states, an HRE will be a finite representation of a set of timed sequences of states.

Let V be a finite set, R^+ be the set of nonnegative real numbers. Each element of V is called a location or state. A finite sequence $(s_1, t_1) \wedge (s_2, t_2) \wedge \dots \wedge (s_m, t_m)$ of elements in $V \times R^+$ is called a timed sequence over V . In this paper, we use \wedge to denote the concatenation of the sequences. The occurrence time $\tau(\sigma)$ of a timed sequence $\sigma = (s_1, t_1) \wedge (s_2, t_2) \wedge \dots \wedge (s_m, t_m)$ over V is defined by $\tau(\sigma) = \sum_{i=1}^m t_i$.

A timed sequence $(s_1, t_1) \wedge (s_2, t_2) \wedge \dots \wedge (s_m, t_m)$ represents a behaviour of a system that the system starts at the state s_1 , stays there for t_1 time units, then changes to s_2 and stays in s_2 for t_2 time units, and so on. The values t_1, t_2, \dots, t_m have to satisfy some time constraints enforced by the system. These time constraints must be incorporated into the finite representation of the system behaviours. By incorporating time constraints into regular expressions, we get Hybrid Regular Expressions.

Definition 1. An HRE \mathcal{R} and the language $\mathcal{L}(\mathcal{R})$ represented by \mathcal{R} over a finite set V of states are defined recursively as follows:

1. ε is an HRE, and $\mathcal{L}(\varepsilon) = \{\varepsilon\}$.
2. If $v \in V$, then v is an HRE, and $\mathcal{L}(v) = \{(v, t) \mid t \in R^+\}$.
3. If \mathcal{R}_1 and \mathcal{R}_2 are HREs, then $\mathcal{R}_1 \wedge \mathcal{R}_2$ is an HRE, and

$$\mathcal{L}(\mathcal{R}_1 \wedge \mathcal{R}_2) = \{\sigma_1 \wedge \sigma_2 \mid \sigma_1 \in \mathcal{L}(\mathcal{R}_1), \sigma_2 \in \mathcal{L}(\mathcal{R}_2)\}.$$

4. If \mathcal{R}_1 and \mathcal{R}_2 are HREs, then $\mathcal{R}_1 \oplus \mathcal{R}_2$ is an HRE, and

$$\mathcal{L}(\mathcal{R}_1 \oplus \mathcal{R}_2) = \mathcal{L}(\mathcal{R}_1) \cup \mathcal{L}(\mathcal{R}_2).$$

5. If \mathcal{R} is an HRE, then \mathcal{R}^* is an HRE, and

$$\mathcal{L}(\mathcal{R}^*) = \{\sigma_1 \wedge \dots \wedge \sigma_m \mid m \geq 0 \text{ and } \bigwedge_{i=1}^m (\sigma_i \in \mathcal{L}(\mathcal{R}))\},$$

where $\sigma_1 \wedge \dots \wedge \sigma_m \hat{=} \varepsilon$ when $m = 0$.

6. If $\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_m$ be HREs, then

$$(\langle \mathcal{R}_1, \lambda_1 \rangle \wedge \langle \mathcal{R}_2, \lambda_2 \rangle \wedge \dots \wedge \langle \mathcal{R}_m, \lambda_m \rangle, \Delta)$$

is an HRE, where Δ be a set of linear inequalities on $\lambda_1, \lambda_2, \dots, \lambda_m$ of the form $a \leq c_1 \lambda_1 + c_2 \lambda_2 + \dots + c_m \lambda_m \leq b$ (a, b , and c_i ($1 \leq i \leq m$) are real numbers) and for any \mathcal{R}_i ($1 \leq i \leq m$) in which there is an occurrence of the combinator $*$, for any $a \leq c_1 \lambda_1 + c_2 \lambda_2 + \dots + c_m \lambda_m \leq b \in \Delta$, if $c_i \neq 0$, then any $c_j \geq 0$ ($1 \leq j \leq m$); and

$$\mathcal{L}((\langle \mathcal{R}_1, \lambda_1 \rangle \wedge \langle \mathcal{R}_2, \lambda_2 \rangle \wedge \dots \wedge \langle \mathcal{R}_m, \lambda_m \rangle, \Delta)) = \left\{ \sigma_1 \wedge \sigma_2 \wedge \dots \wedge \sigma_m \left| \begin{array}{l} \text{each } \sigma_i \text{ } (1 \leq i \leq m) \text{ belongs to } \mathcal{L}(\mathcal{R}_i) \text{ such that} \\ \text{for all } a \leq \sum_{i=1}^m c_i \lambda_i \leq b \in \Delta, a \leq \sum_{i=1}^m c_i \tau(\sigma_i) \leq b \end{array} \right. \right\}.$$

When $\Delta = \{a \leq \sum_{i=1}^m \lambda_i \leq b\}$, $(\langle \mathcal{R}_1, \lambda_1 \rangle \wedge \langle \mathcal{R}_2, \lambda_2 \rangle \wedge \dots \wedge \langle \mathcal{R}_m, \lambda_m \rangle, \Delta)$ is taken to be $(\mathcal{R}_1 \wedge \mathcal{R}_2 \wedge \dots \wedge \mathcal{R}_m, [a, b])$. \square

Although the traditional regular expressions are powerful enough to describe the behaviour of finite automata, it is not the case for HREs to describe the behaviour of all linear hybrid automata. Nevertheless, HRE is simple and powerful enough to express the real-time behaviour of many real-time hybrid systems encountered in practice.

For example, the behaviour of the linear hybrid automaton (Fig. 1) modelling the water level monitor in the introduction can be represented by the following HRE \mathcal{R}_w :

$$\mathcal{R}_w = \varepsilon \oplus (s_1, [0, 9]) \oplus (s_1, [9, 9]) \wedge (s_2, [0, 2]) \oplus \mathcal{R}_1 \\ \oplus \mathcal{R}_2 \wedge \mathcal{R}_3^* \wedge ((s_3, [0, 2]) \oplus \mathcal{R}_4 \oplus \mathcal{R}_5 \oplus \mathcal{R}_6)$$

where

$$\begin{aligned} \mathcal{R}_1 &= (\langle s_1, \lambda_1 \rangle \wedge \langle s_2, \lambda_2 \rangle \wedge \langle s_4, \lambda_3 \rangle, \{\lambda_1 = 9, \lambda_2 = 2, 2\lambda_3 - \lambda_2 \leq 5\}) \\ \mathcal{R}_2 &= (\langle s_1, \lambda_1 \rangle \wedge \langle s_2, \lambda_2 \rangle \wedge \langle s_4, \lambda_3 \rangle, \{\lambda_1 = 9, \lambda_2 = 2, 2\lambda_3 - \lambda_2 = 5\}) \\ \mathcal{R}_3 &= (\langle s_3, \lambda_1 \rangle \wedge \langle s_1, \lambda_2 \rangle \wedge \langle s_2, \lambda_3 \rangle \wedge \langle s_4, \lambda_4 \rangle, \\ &\quad \{\lambda_1 = 2, \lambda_2 - 2\lambda_1 = 5, \lambda_3 = 2, 2\lambda_4 - \lambda_3 = 5\}) \\ \mathcal{R}_4 &= (\langle s_3, \lambda_1 \rangle \wedge \langle s_1, \lambda_2 \rangle, \{\lambda_1 = 2, \lambda_2 - 2\lambda_1 \leq 5\}) \\ \mathcal{R}_5 &= (\langle s_3, \lambda_1 \rangle \wedge \langle s_1, \lambda_2 \rangle \wedge \langle s_2, \lambda_3 \rangle, \{\lambda_1 = 2, \lambda_2 - 2\lambda_1 = 5, 0 \leq \lambda_3 \leq 2\}) \\ \mathcal{R}_6 &= (\langle s_3, \lambda_1 \rangle \wedge \langle s_1, \lambda_2 \rangle \wedge \langle s_2, \lambda_3 \rangle \wedge \langle s_4, \lambda_4 \rangle, \\ &\quad \{\lambda_1 = 2, \lambda_2 - 2\lambda_1 = 5, \lambda_3 = 2, -5 \leq \lambda_3 - 2\lambda_4\}). \end{aligned}$$

Since HREs form a very simple formalism to model real-time and hybrid systems, hopefully many problems are decidable for the class of real-time and hybrid systems defined by HREs. In next two sections, we show that for the class of linear hybrid automata whose behaviour can be represented by HREs, two class of reachability problems and the satisfaction problem for linear duration invariants are decidable. In the rest of this section, we introduce some concepts concerning HREs that will be used in next two sections.

For an HRE \mathcal{R} , if $\mathcal{L}(\mathcal{R}) = \emptyset$, then \mathcal{R} is said to be *empty*.

Definition 2. For an HRE \mathcal{R} , its *sub-expressions* are defined recursively by:

1. \mathcal{R} is a sub-expression of \mathcal{R} .
2. If $\mathcal{R} = \mathcal{R}_1 \wedge \mathcal{R}_2$ or $\mathcal{R} = \mathcal{R}_1 \oplus \mathcal{R}_2$, where \mathcal{R}_1 and \mathcal{R}_2 are HREs, then all the sub-expressions of \mathcal{R}_1 and \mathcal{R}_2 are sub-expressions of \mathcal{R} .
3. If $\mathcal{R} = \mathcal{R}_1^*$ or $\mathcal{R} = (\mathcal{R}_1, [a, b])$, where \mathcal{R}_1 is an HRE, then all the sub-expressions of \mathcal{R}_1 are sub-expressions of \mathcal{R} .
4. If $\mathcal{R} = (\langle \mathcal{R}_1, \lambda_1 \rangle \wedge \langle \mathcal{R}_2, \lambda_2 \rangle \wedge \dots \wedge \langle \mathcal{R}_m, \lambda_m \rangle, \Delta)$, where each \mathcal{R}_i ($1 \leq i \leq m$) is an HRE, then all the sub-expressions of \mathcal{R}_i are sub-expressions of \mathcal{R} . \square

A *simple* HRE is an HRE in which there is no occurrence of the combinators $*$ (repetition) and \oplus (union). From Definition 1, any simple HRE \mathcal{R} can be rewritten as a simple HRE \mathcal{R}' of the form $(\langle v_1, \lambda_1 \rangle \wedge \langle v_2, \lambda_2 \rangle \wedge \dots \wedge \langle v_m, \lambda_m \rangle, \Delta)$ such that $\mathcal{L}(\mathcal{R}) = \mathcal{L}(\mathcal{R}')$, where for each i ($1 \leq i \leq m$), $v_i \in V$. Therefore, from now on, we assume that any simple HRE is of the form

$$(\langle v_1, \lambda_1 \rangle \wedge \langle v_2, \lambda_2 \rangle \wedge \dots \wedge \langle v_m, \lambda_m \rangle, \Delta),$$

where Δ is a finite set of linear inequalities of the form $a \leq \sum_{i=1}^m c_i \lambda_i \leq b$.

For any simple HRE $\mathcal{R} = (\langle v_1, \lambda_1 \rangle \wedge \langle v_2, \lambda_2 \rangle \wedge \dots \wedge \langle v_m, \lambda_m \rangle, \Delta)$, let $M_\tau(\mathcal{R})$ ($m_\tau(\mathcal{R})$) denote the supremum (infimum) of the set $\{\tau(\sigma) \mid \sigma \in \mathcal{L}(\mathcal{R})\}$. $M_\tau(\mathcal{R})$ ($m_\tau(\mathcal{R})$) can be calculated by finding the maximal (minimal) value of the linear objective function $\lambda_1 + \lambda_2 + \dots + \lambda_m$ subject to the group of linear inequalities in Δ , which is a classical linear programming problem. If $m_\tau(\mathcal{R}) = 0$, \mathcal{R} is said to be a *zero-simple* HRE; otherwise \mathcal{R} is said to be a *nonzero-simple* HRE.

By a *normal form* we mean an HRE of the form $\mathcal{R}_1 \oplus \mathcal{R}_2 \oplus \dots \oplus \mathcal{R}_m$, where \mathcal{R}_j s are simple HREs.

Let \mathcal{R} be an HRE, and \mathcal{R}_1 be a sub-expression of \mathcal{R} . Replacing an occurrence of \mathcal{R}_1 in \mathcal{R} with a letter X , we obtain a *context* of X . Any context $\mathcal{C}(X)$ of X , is associated with two real numbers $\varphi(\mathcal{C}(X))$ and $\omega(\mathcal{C}(X))$, which specify a lower bound and an upper bound of the constraints on the occurrence time enforced by the context on the variable X . If the context does not enforce any time constraint on X then $\varphi(\mathcal{C}(X)) = 0$ and $\omega(\mathcal{C}(X)) = \infty$.

Definition 3. A context $\mathcal{C}(X)$ of X , $\varphi(\mathcal{C}(X))$ and $\omega(\mathcal{C}(X))$ are defined recursively as:

1. X is a context of X , and $\varphi(X) = 0$ and $\omega(X) = \infty$ (no additional constraint).
2. If $\mathcal{C}_1(X)$ is a context of X and \mathcal{R} is an HRE, then $\mathcal{C}(X) = \mathcal{C}_1(X) \wedge \mathcal{R}$ and $\mathcal{C}(X) = \mathcal{R} \wedge \mathcal{C}_1(X)$ are contexts of X , and

$$\varphi(\mathcal{C}(X)) = \varphi(\mathcal{C}_1(X)), \omega(\mathcal{C}(X)) = \omega(\mathcal{C}_1(X))$$

(no additional constraint).

3. If $\mathcal{C}_1(X)$ is a context of X and \mathcal{R} is an HRE, then $\mathcal{C}(X) = \mathcal{C}_1(X) \oplus \mathcal{R}$ and $\mathcal{C}(X) = \mathcal{R} \oplus \mathcal{C}_1(X)$ are contexts of X , and

$$\varphi(\mathcal{C}(X)) = \varphi(\mathcal{C}_1(X)), \omega(\mathcal{C}(X)) = \omega(\mathcal{C}_1(X))$$

(no additional constraint).

4. If $\mathcal{C}_1(X)$ is a context of X , then $\mathcal{C}(X) = \mathcal{C}_1(X)^*$ is a context of X , and

$$\varphi(\mathcal{C}(X)) = \varphi(\mathcal{C}_1(X)), \omega(\mathcal{C}(X)) = \omega(\mathcal{C}_1(X))$$

(no additional constraint).

5. If $\mathcal{C}_1(X)$ is a context of X and $\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_m$ are HREs, then

$$\mathcal{C}(X) = \left(\langle \mathcal{R}_1, \lambda_1 \rangle \wedge \langle \mathcal{R}_2, \lambda_2 \rangle \wedge \dots \wedge \langle \mathcal{R}_{k-1}, \lambda_{k-1} \rangle \wedge \langle \mathcal{C}_1(X), \lambda_k \rangle \wedge \langle \mathcal{R}_{k+1}, \lambda_{k+1} \rangle \wedge \dots \wedge \langle \mathcal{R}_m, \lambda_m \rangle, \Delta \right)$$

is a context of X ($1 \leq k \leq m$). Let a be the maximal value of the set $\{a/c_k \mid a \leq c_1\lambda_1 + c_2\lambda_2 + \dots + c_k\lambda_k + \dots + c_m\lambda_m \leq b \in \Delta \text{ and } c_k \neq 0\}$, and b be the maximal value of the linear function λ_k subject to all the linear inequalities in Δ respectively. Then

$$\varphi(\mathcal{C}(X)) = \max(\varphi(\mathcal{C}_1(X)), a), \omega(\mathcal{C}(X)) = \min(\omega(\mathcal{C}_1(X)), b)$$

(additional constraint enforced by Δ).

□

For any context $\mathcal{C}(X)$, replacing X in $\mathcal{C}(X)$ with an HRE, say \mathcal{R} , we obtain an HRE, denoted by $\mathcal{C}(\mathcal{R})$.

3 Checking Linear Hybrid Automata for Reachability

The reachability problem is central to the verification of hybrid systems. In general, the reachability problem for linear hybrid systems is undecidable [2,5,7]. But the following two class of reachability problem is decidable for the class of linear hybrid automata whose behaviour can be expressed by HREs.

The one is a typical reachability problem studied in [5]: Given a final location s , is there a behaviour of the automaton terminating at location s . Suppose \mathcal{R} is an HRE representing the behaviour of the automaton terminating at location s . The reachability problem can be solved by checking the emptiness of \mathcal{R} .

The other is called time-bounded reachability problem in [6]: Given a final location s and a time interval $[a, b]$, is there a behaviour of the automaton terminating at location s such that the total elapsed time of the behaviour is in the time interval $[a, b]$. Let \mathcal{R} is an HRE representing the behaviour of the automaton terminating at location s . The reachability problem is equivalent to the problem of checking the emptiness of $(\mathcal{R}, [a, b])$.

In the following, we solve the problem checking the emptiness of an HRE. By Definition 1 and 2, it is not difficult to prove the following two theorems.

Theorem 1. For an empty HRE \mathcal{R} and for an HRE \mathcal{R}_1 ,

$$\begin{aligned} \mathcal{L}(\mathcal{R} \wedge \mathcal{R}_1) &= \mathcal{L}(\mathcal{R}_1 \wedge \mathcal{R}) = \emptyset, \\ \mathcal{L}(\mathcal{R} \oplus \mathcal{R}_1) &= \mathcal{L}(\mathcal{R}_1 \oplus \mathcal{R}) = \mathcal{L}(\mathcal{R}_1), \\ \mathcal{L}(\mathcal{R}^*) &= \{\epsilon\}, \text{ and } \mathcal{L}(\mathcal{R}, [a, b]) = \emptyset. \end{aligned}$$

For an HRE \mathcal{R} which has the form $(\langle \mathcal{R}_1, \lambda_1 \rangle \wedge \langle \mathcal{R}_2, \lambda_2 \rangle \wedge \dots \wedge \langle \mathcal{R}_m, \lambda_m \rangle, \Delta)$, if the group of inequalities has no solution or there is \mathcal{R}_i ($1 \leq i \leq m$) such that $\mathcal{L}(\mathcal{R}_i) = \emptyset$, then $\mathcal{L}(\mathcal{R}) = \emptyset$. □

Theorem 2. Let \mathcal{R} be an HRE and \mathcal{R}_1 be a sub-expression of \mathcal{R} . Let \mathcal{R}'_1 be an HRE such that $\mathcal{L}(\mathcal{R}'_1) = \mathcal{L}(\mathcal{R}_1)$. Suppose \mathcal{R}' is constructed from replacing an occurrence of \mathcal{R}_1 in \mathcal{R} with \mathcal{R}'_1 . Then, $\mathcal{L}(\mathcal{R}) = \mathcal{L}(\mathcal{R}')$. \square

By Theorem 1 and 2, for any HRE \mathcal{R} , we can find out an HRE \mathcal{R}' such that

- $\mathcal{L}(\mathcal{R}) = \mathcal{L}(\mathcal{R}')$, and
- \mathcal{R}' has no sub-expression which is of the form

$$(\langle \mathcal{R}_1, \lambda_1 \rangle \wedge \langle \mathcal{R}_2, \lambda_2 \rangle \wedge \dots \wedge \langle \mathcal{R}_m, \lambda_m \rangle, \Delta).$$

where the group of inequalities in Δ has no solution.

Hence, for simplicity, from now on, unless otherwise stated, we assume that all HREs under consideration have no sub-expression which is of the form

$$(\langle \mathcal{R}_1, \lambda_1 \rangle \wedge \langle \mathcal{R}_2, \lambda_2 \rangle \wedge \dots \wedge \langle \mathcal{R}_m, \lambda_m \rangle, \Delta)$$

where the group of inequalities in Δ has no solution.

First, let us consider the problem checking if a simple HRE is empty. Let \mathcal{R} be a simple HRE $\mathcal{R} = (\langle v_1, \lambda_1 \rangle \wedge \langle v_2, \lambda_2 \rangle \wedge \dots \wedge \langle v_m, \lambda_m \rangle, \Delta)$. From the definition of HREs, every $\sigma \in \mathcal{L}(\mathcal{R})$ is of the form $(v_1, t_1) \wedge (v_2, t_2) \wedge \dots \wedge (v_m, t_m)$, where t_1, t_2, \dots, t_m satisfy the group of linear inequalities represented by Δ . Hence, the problem checking the emptiness of a simple HRE can be solved by checking if the group of linear inequalities in Δ has no solution, which can be solved by linear programming.

Let $\mathcal{N} = \mathcal{R}_1 \oplus \mathcal{R}_2 \oplus \dots \oplus \mathcal{R}_m$ be a normal form. Hence, each \mathcal{R}_i ($1 \leq i \leq m$) is a simple HRE. Since $\mathcal{L}(\mathcal{N}) = \emptyset \Leftrightarrow \bigwedge_{i=1}^m \mathcal{L}(\mathcal{R}_i) = \emptyset$, the problem of checking if \mathcal{N} is empty can be solved by solving m linear programming problems checking if \mathcal{R}_i is empty, $i = 1, 2, \dots, m$.

Therefore, for a general HRE \mathcal{R} , if we can effectively find a normal form \mathcal{N} such that $\mathcal{L}(\mathcal{R}) = \emptyset$ if and only if $\mathcal{L}(\mathcal{N}) = \emptyset$, then we can check if \mathcal{R} is empty effectively.

For an HRE \mathcal{R} , we attempt to find a normal form \mathcal{N} such that $\mathcal{L}(\mathcal{R}) = \emptyset$ if and only if $\mathcal{L}(\mathcal{N}) = \emptyset$ by the following procedure:

Step 0. Let $\mathcal{R}' := \mathcal{R}$.

Step 1. For \mathcal{R}' , distributing \wedge over \oplus , and $[a, b]$ over \oplus , we obtain \mathcal{Q} . If \mathcal{Q} is a normal form, then we are done.

Step 2. For a sub-expression \mathcal{Q}_S of \mathcal{Q} which is of the form $\mathcal{Q}_S = \mathcal{Q}_1^*$, replacing an occurrence of \mathcal{Q}_S in \mathcal{Q} with X , we obtain a context $\mathcal{C}_{\mathcal{Q}}(X)$ such that $\mathcal{R}' = \mathcal{C}_{\mathcal{Q}}(\mathcal{Q}_S)$.

Step 3. Finding an HRE \mathcal{Q}'_S in which there is no occurrence of combinator $*$ such that $\mathcal{C}_{\mathcal{Q}}(\mathcal{Q}_S) = \emptyset$ if and only if $\mathcal{C}_{\mathcal{Q}}(\mathcal{Q}'_S) = \emptyset$. Let $\mathcal{R}' := \mathcal{C}_{\mathcal{Q}}(\mathcal{Q}'_S)$, and go to Step 1. \square

Obviously the procedure is correct. The problem is how to find \mathcal{Q}'_S in Step 3. The following lemmas and theorems will help to solve that problem.

Let $\mathcal{C}(X)$ be a context. For a real number x , let $\lfloor x \rfloor$ denote the floor of x . For an HRE \mathcal{R} , let \mathcal{R}^j denote the j -repetition of \mathcal{R}

$$\mathcal{R}^j = \underbrace{\mathcal{R} \wedge \mathcal{R} \wedge \dots \wedge \mathcal{R}}_j, \mathcal{R}^0 = \varepsilon.$$

Lemma 1. Let \mathcal{R} and \mathcal{R}' be HREs. If for any $\sigma \in \mathcal{L}(\mathcal{R})$, there is $\sigma' \in \mathcal{L}(\mathcal{R}')$ such that $\tau(\sigma) = \tau(\sigma')$, then $\mathcal{L}(\mathcal{C}(\mathcal{R}')) = \emptyset$ implies $\mathcal{L}(\mathcal{C}(\mathcal{R})) = \emptyset$. \square

Lemma 2. Suppose $\omega(\mathcal{C}(X)) = \infty$, and \mathcal{R} be a nonzero-simple HRE \mathcal{R} . Then for any real number a , for any $\sigma \in \mathcal{L}(\mathcal{C}(\mathcal{R}^*))$ such that $\tau(\sigma) \geq a$, there is $\sigma' \in \mathcal{L}(\mathcal{C}(\oplus_{j=0}^p \mathcal{R}^j))$ such that $\tau(\sigma') \geq a$, where $p = (\lfloor \max(\varphi(\mathcal{C}(X), a) / m_\tau(\mathcal{R}) \rfloor + 1)$. \square

Lemma 3. Let \mathcal{R} be a nonzero-simple HRE. Then $\mathcal{L}(\mathcal{C}(\oplus_{j=0}^p \mathcal{R}^j)) \supseteq \mathcal{L}(\mathcal{C}(\mathcal{R}^*))$, where $p = \lfloor \omega(\mathcal{C}(X)) / m_\tau(\mathcal{R}) \rfloor + 1$. \square

These lemmas can be proved by induction on the structure of context. Their detailed proofs are omitted because of space considerations. From these lemmas, we can prove the following theorems.

Theorem 3. Let \mathcal{R}_1 and \mathcal{R}_2 be HREs. Then

$$\mathcal{L}(\mathcal{C}((\mathcal{R}_1 \oplus \mathcal{R}_2)^*)) = \emptyset \text{ iff } \mathcal{L}(\mathcal{C}((\mathcal{R}_1^*) \wedge (\mathcal{R}_2^*))) = \emptyset.$$

Proof. By Definition 1, $\mathcal{L}((\mathcal{R}_1^*) \wedge (\mathcal{R}_2^*)) \subseteq \mathcal{L}((\mathcal{R}_1 \oplus \mathcal{R}_2)^*)$. From Lemma 1, the half of the claim follows, i.e.

$$\mathcal{L}(\mathcal{C}((\mathcal{R}_1 \oplus \mathcal{R}_2)^*)) = \emptyset \text{ implies } \mathcal{L}(\mathcal{C}((\mathcal{R}_1^*) \wedge (\mathcal{R}_2^*))) = \emptyset.$$

The other half can be proved as follows. Since any $\sigma \in \mathcal{L}(\mathcal{C}((\mathcal{R}_1 \oplus \mathcal{R}_2)^*))$ can be permuted into $\sigma' \in \mathcal{L}(\mathcal{C}((\mathcal{R}_1^*) \wedge (\mathcal{R}_2^*)))$, from lemma 1, the result follows. \square

Theorem 4. Let $\mathcal{R} = (\langle v_1, \lambda_1 \rangle \wedge \langle v_2, \lambda_2 \rangle \wedge \dots \wedge \langle v_m, \lambda \rangle, \Delta)$ be a zero-simple HRE. Let Δ' be the set

$$\{0 \leq \sum_{i=1}^m c_i \lambda_i \mid 0 \leq \sum_{i=1}^m c_i \lambda_i \leq b \in \Delta \wedge \exists j \cdot (1 \leq j \leq m \wedge c_j < 0)\},$$

and $\mathcal{R}' = (\langle v_1, \lambda_1 \rangle \wedge \langle v_2, \lambda_2 \rangle \wedge \dots \wedge \langle v_m, \lambda \rangle, \Delta')$. Then

$$\mathcal{L}(\mathcal{C}(\mathcal{R}^*)) = \emptyset \text{ iff } \mathcal{L}(\mathcal{C}(\mathcal{R}')) = \emptyset.$$

Proof. Before the proof, we should note that by the definition of zero-simple HREs, $\tau(\mathcal{R}) = 0$ implies that for any inequality $a \leq c_1 \lambda_1 + c_2 \lambda_2 + \dots + c_m \lambda_m \leq b$ in Δ , $a \leq 0$ and $b \geq 0$.

The half of the claim that $\mathcal{L}(\mathcal{C}(\mathcal{R}')) = \emptyset$ implies $\mathcal{L}(\mathcal{C}(\mathcal{R}^*)) = \emptyset$, is explained as follows. By Definition 1, any $\sigma \in \mathcal{L}(\mathcal{R}^*)$ is of the form $\sigma_1 \wedge \sigma_2 \wedge \dots \wedge \sigma_n$, where

$$\sigma_i = (v_1, t_{i1}) \wedge (v_2, t_{i2}) \wedge \dots \wedge (v_m, t_{im}) \in \mathcal{L}(\mathcal{R}) \quad (i = 1, 2, \dots, n).$$

For any j ($1 \leq j \leq m$), let $t'_j = t_{1j} + t_{2j} + \dots + t_{nj}$, and let

$$\sigma' = (v_1, t'_1) \wedge (v_2, t'_2) \wedge \dots \wedge (v_m, t'_m).$$

Since for any i ($1 \leq i \leq n$), $t_{i1}, t_{i2}, \dots, t_{im}$ satisfy Δ , t'_1, t'_2, \dots, t'_m satisfy Δ' as well. It follows that $\sigma' \in \mathcal{L}(\mathcal{R}')$. Since $\tau(\sigma) = \tau(\sigma')$, the first half of the claim follows from Lemma 1.

The other half of the claim, i.e. $\mathcal{L}(\mathcal{C}(\mathcal{R}^*)) = \emptyset$ implies $\mathcal{L}(\mathcal{C}(\mathcal{R}')) = \emptyset$, can be proved as follows. For any $\sigma' = (v_1, t'_1) \wedge (v_2, t'_2) \wedge \dots \wedge (v_m, t'_m) \in \mathcal{L}(\mathcal{R}')$, since t_1, t_2, \dots, t_m satisfy Δ' , for any $0 \leq \sum_{i=1}^m c_i \lambda_i \leq b \in \Delta$, we have $\sum_{i=1}^m c_i t_i \geq 0$. Because for each inequality $a \leq c_1 \lambda_1 + c_2 \lambda_2 + \dots + c_m \lambda_m \leq b$ in Δ , $a \leq 0$ and $b \geq 0$, and because Δ is a finite set, we can choose a natural number p such that for any inequality $a \leq c_1 \lambda_1 + c_2 \lambda_2 + \dots + c_m \lambda_m \leq b \in \Delta$,

$$a \leq \frac{c_1 t_1 + c_2 t_2 + \dots + c_m t_m}{p} \leq b.$$

For each i ($1 \leq i \leq m$), let $b_i = t_i/p$, and let $\sigma_b = (v_1, b_1) \wedge (v_2, b_2) \wedge \dots \wedge (v_m, b_m)$. Obviously, $\sigma \in \mathcal{L}(\mathcal{R})$. Let

$$\sigma = \underbrace{\sigma_b \wedge \sigma_b \wedge \dots \wedge \sigma_b}_p.$$

It follows that $\sigma \in \mathcal{L}(\mathcal{R}^*)$. Since $\tau(\sigma) = \tau(\sigma')$, by Lemma 1, $\mathcal{L}(\mathcal{C}(\mathcal{R}^*)) = \emptyset$ implies $\mathcal{L}(\mathcal{C}(\mathcal{R}')) = \emptyset$. \square

Theorem 5. Suppose $\omega(\mathcal{C}(X)) = \infty$, and \mathcal{R} be a nonzero-simple HRE. Then

$$\mathcal{L}(\mathcal{C}(\mathcal{R}^*)) = \emptyset \text{ iff } \mathcal{L}(\mathcal{C}(\oplus_{j=0}^p \mathcal{R}^j)) = \emptyset,$$

where $p = (\lfloor \varphi(\mathcal{C}(X)) / m_\tau(\mathcal{R}) \rfloor + 1)$.

Proof. By Definition 1, $\mathcal{L}(\mathcal{R}^*) \supseteq \mathcal{L}(\oplus_{j=0}^p \mathcal{R}^j)$ holds, which by Lemma 1 implies a half of the claim, i.e. $\mathcal{L}(\mathcal{C}(\mathcal{R}^*)) = \emptyset$ implies $\mathcal{L}(\mathcal{C}(\oplus_{j=0}^p \mathcal{R}^j)) = \emptyset$. The other half is straightforward from Lemma 2. \square

Theorem 6. Suppose $\omega(\mathcal{C}(X)) \neq \infty$, and \mathcal{R} be a nonzero-simple HRE. Then

$$\mathcal{L}(\mathcal{C}(\mathcal{R}^*)) = \emptyset \text{ iff } \mathcal{L}(\mathcal{C}(\oplus_{j=0}^p \mathcal{R}^j)) = \emptyset,$$

where $p = (\lfloor \omega(\mathcal{C}(X)) / m_\tau(\mathcal{R}) \rfloor + 1)$.

Proof. By Definition 1, $\mathcal{L}(\mathcal{R}^*) \supseteq \mathcal{L}(\oplus_{j=0}^p \mathcal{R}^j)$ holds, which by Lemma 1 implies a half of the claim, i.e. $\mathcal{L}(\mathcal{C}(\mathcal{R}^*)) = \emptyset$ implies $\mathcal{L}(\mathcal{C}(\oplus_{j=0}^p \mathcal{R}^j)) = \emptyset$. The other half is straightforward from Lemma 3. \square

Based on the above theorems, the algorithm to check an HRE \mathcal{R} for emptiness is now described as follows.

Step 0. Let $\mathcal{R}' := \mathcal{R}$.

Step 1. For \mathcal{R}' , distributing \wedge over \oplus , and $[a, b]$ over \oplus , we obtain \mathcal{Q} .

Step 2. Finding a sub-expression \mathcal{Q}_S of \mathcal{Q} which has one of the following three forms:

1. $\mathcal{Q}_S = (\mathcal{R}_1 \oplus \mathcal{R}_2 \oplus \dots \oplus \mathcal{R}_k)^*$ ($k \geq 2$), where every \mathcal{R}_i ($1 \leq i \leq m$) is a simple HRE.
2. $\mathcal{Q}_S = \mathcal{R}_1^*$, where \mathcal{R}_1 is a nonzero-simple HRE.
3. $\mathcal{Q}_S = \mathcal{R}_1^*$, where \mathcal{R}_1 is a zero-simple HRE.

If such \mathcal{Q}_S could not be found, goto Step 6 (note that it is not difficult to prove that if we can not find out such a \mathcal{Q}_S , then \mathcal{Q} is a normal form); otherwise replacing the occurrence of \mathcal{Q}_S in \mathcal{Q} with X , we get a context $\mathcal{C}_Q(X)$ such that $\mathcal{Q} = \mathcal{C}_Q(\mathcal{Q}_S)$. Then, if \mathcal{Q}_S has the first form, goto Step 3; if \mathcal{Q}_S has the second form, goto Step 4; if \mathcal{Q}_S has the third form, goto Step 5.

Step 3. By Theorem 3, we transform \mathcal{Q} into $\mathcal{Q}' = \mathcal{C}_Q((\mathcal{R}_1)^* \wedge (\mathcal{R}_2)^* \wedge \dots \wedge (\mathcal{R}_m)^*)$. Then, let $\mathcal{R}' := \mathcal{Q}'$, and goto Step 1.

Step 4. We first calculate $\omega(\mathcal{C}_Q(X))$. If $\omega(\mathcal{C}_Q(X)) \neq \infty$, then by Theorem 6, we transform \mathcal{Q} into $\mathcal{Q}' = \mathcal{C}_Q(\oplus_{j=0}^p \mathcal{R}_1^j)$, where $p = (\lfloor \omega(\mathcal{C}_Q(X)) / m_\tau(\mathcal{R}_1) \rfloor + 1)$. Let $\mathcal{R}' := \mathcal{Q}'$, and goto Step 1.

Otherwise, $\omega(\mathcal{C}_Q(X)) = \infty$. By Theorem 5, we transform \mathcal{Q} into $\mathcal{Q}' = \mathcal{C}_Q(\oplus_{j=0}^p \mathcal{R}_1^j)$, where $p = (\lfloor \varphi(\mathcal{C}_Q(X)) / m_\tau(\mathcal{R}_1) \rfloor + 1)$. Let $\mathcal{R}' := \mathcal{Q}'$, and goto Step 1.

Step 5. By Theorem 4, we transform \mathcal{Q} into $\mathcal{Q}' = \mathcal{C}_Q(\mathcal{R}'_1)$, where \mathcal{R}'_1 is the simple HRE defined from \mathcal{R}_1 in Theorem 4. Let $\mathcal{R}' := \mathcal{Q}'$, and goto Step 1.

Step 6. Since \mathcal{Q} is a normal form now, we check the emptiness of \mathcal{Q} by linear programming. If \mathcal{Q} is empty, then \mathcal{R} is empty; otherwise \mathcal{R} is not empty. \square

4 Checking Linear Hybrid Automata for Linear Duration Invariants

In this section, the problem we are concerned can be described as follows: Given a hybrid automaton A , given a linear duration invariant \mathcal{D} , decide efficiently whether A satisfy \mathcal{D} .

4.1 Linear Duration Invariants

Linear duration invariants [4] are constructed from linear inequalities of integrated durations of system states. They form an important class of Duration Calculus (DC) [3] formulas. In DC, states are modelled as Boolean functions from reals (representing continuous time) to $\{0, 1\}$, where 1 denotes state presence, and 0 denotes state absence. For a state S , the interval variable $\int S$ of DC is a function from bounded and closed intervals to reals which stands for the accumulated presence time (duration) of state S over the intervals, and is

defined formally by $\int S[a, b] \triangleq \int_a^b S(t) dt$, where $[a, b]$ ($b \geq a$) is a bounded interval of time. A linear duration invariant \mathcal{D} in DC is of the form

$$T \geq \int 1 \geq t \Rightarrow \bigwedge_{j=1}^k \left(\sum_{i=1}^n c_{ij} \int S_i \leq M_j \right),$$

where T, t, c_{ij}, M_j are real numbers (T may be ∞).

The meaning of a linear duration invariant \mathcal{D} is that: if the system is observed for an interval of time satisfying the premise of \mathcal{D} , then the duration of the system states must satisfy the consequence of \mathcal{D} . It turns out that many real-time properties can be written as a linear duration invariant.

For example, the requirement of the water-level monitor in Fig.1, which is that the monitor must keep the water level in between 1 and 12 inches, can be expressed by linear duration invariants as well. We know that when the control is in locations s_1 or s_2 , the water level rises 1 inch per second, and when the control is in locations s_3 or s_4 , the water level falls by 2 inch per second. Furthermore, for an interval $[0, t]$, the accumulated time that the system stays in s_1 or s_2 is $\int s_1 + \int s_2$, and the accumulated time that the system stays in s_3 or s_4 is $\int s_3 + \int s_4$. Therefore, the water level at time t , given that at the beginning the water level is one inch, is $1 + \int s_1 + \int s_2 - 2(\int s_3 + \int s_4)$. Hence, the requirement for the water-level monitor can be described by the following linear duration invariants

$$\begin{aligned} 0 \leq \int 1 \leq \infty &\Rightarrow 1 + \int s_1 + \int s_2 - 2(\int s_3 + \int s_4) \leq 12; \\ 0 \leq \int 1 \leq \infty &\Rightarrow 1 + \int s_1 + \int s_2 - 2(\int s_3 + \int s_4) \geq 1. \end{aligned}$$

For a location $v \in V$, for a predicate S over V , let $v \Rightarrow S$ denote that S holds during the system stays at v . A timed sequence $\sigma = (v_1, t_1) \wedge (v_2, t_2) \wedge \dots \wedge (v_m, t_m)$ over V satisfies a linear duration invariant \mathcal{D} iff $\bigwedge_{j=1}^k \left(\sum_{i=1}^n c_{ij} \left(\sum_{u \in \alpha_i} t_u \right) \leq M_j \right)$ when $T \geq \sum_{u=1}^m t_u \geq t$, where $\alpha_i \triangleq \{u \mid (1 \leq u \leq m) \wedge (v_u \Rightarrow S_i)\}$. A hybrid automaton satisfies a linear duration invariant if and only if every behaviour of the automaton satisfies the linear duration invariant. An HRE \mathcal{R} satisfies a linear duration invariant \mathcal{D} , denoted by $\mathcal{R} \models \mathcal{D}$, iff any timed sequences $\sigma \in \mathcal{L}(\mathcal{R})$ satisfies \mathcal{D} .

4.2 Checking HREs for Linear Duration Invariants

Now, we consider the problem checking an HRE \mathcal{R} for linear duration invariant \mathcal{D} . Without loss of generality, throughout this section, let \mathcal{D} be

$$t \leq \int 1 \leq T \Rightarrow \sum_{i=1}^n c_i \int S_i \leq M,$$

and for any $\sigma = (v_1, t_1) \wedge (v_2, t_2) \wedge \dots \wedge (v_m, t_m) \in \mathcal{L}(\mathcal{R})$, let $\theta(\sigma, \mathcal{D})$ be the value of $\sum_{i=1}^n c_i \int S_i$ evaluated over σ ,

$$\theta(\sigma, \mathcal{D}) = \sum_{i=1}^n c_i \left(\sum_{u \in \alpha_i} t_u \right),$$

where $\alpha_i = \{u \mid (1 \leq u \leq m) \wedge (v_u \Rightarrow S_i)\}$.

For simplicity, we assume that all HREs under consideration are not empty and do not have any empty sub-expression.

For any nonzero-simple HRE $\mathcal{R} = (\langle v_1, \lambda_1 \rangle \wedge \langle v_2, \lambda_2 \rangle \wedge \dots \wedge \langle v_m, \lambda_m \rangle, \Delta)$, let $M_\theta(\mathcal{R})$ denote the supremum of the set $\{\theta(\sigma, \mathcal{D}) \mid \sigma \in \mathcal{L}(\mathcal{R})\}$. $M_\tau(\mathcal{R})$, $M_\theta(\mathcal{R})$ can be calculated effectively by finding the maximal value of the linear objective function $\sum_{i=1}^n c_i (\sum_{u \in \alpha_i} \lambda_u)$ subject to the group of linear inequalities in Δ .

Let \mathcal{R} be a simple HRE $\mathcal{R} = (\langle v_1, \lambda_1 \rangle \wedge \langle v_2, \lambda_2 \rangle \wedge \dots \wedge \langle v_m, \lambda_m \rangle, \Delta)$. From the definition of HREs, every $\sigma \in \mathcal{L}(\mathcal{R})$ is of the form

$$(v_1, t_1) \wedge (v_2, t_2) \wedge \dots \wedge (v_m, t_m),$$

where t_1, t_2, \dots, t_m satisfy the group of linear inequalities represented by Δ . Denoting this group of linear inequalities by C_1 , the problem of checking $\mathcal{R} \models \mathcal{D}$ is then equivalent to the problem of finding the maximum value of the linear function $\sum_{i=1}^n c_{ij} (\sum_{u \in \alpha_i} t_u)$ subject to the linear constraints C_1 and C_2 and checking whether it is not greater than M_j for all $j = 1, \dots, k$, where C_2 denotes the inequality

$$t \leq t_1 + t_2 + \dots + t_m \leq T.$$

The latter are linear programming problems.

Let $\mathcal{N} = \mathcal{R}_1 \oplus \mathcal{R}_2 \oplus \dots \oplus \mathcal{R}_m$ be a normal form. Hence, each \mathcal{R}_i ($1 \leq i \leq m$) is a simple HRE. Since, by Definition 3,

$$\mathcal{N} \models \mathcal{D} \Leftrightarrow \bigwedge_{i=1}^m \mathcal{R}_i \models \mathcal{D},$$

the problem of checking \mathcal{N} for \mathcal{D} can be solved by solving m linear programming problems $\mathcal{R}_i \models \mathcal{D}$, $i = 1, 2, \dots, m$.

Therefore, for a general HRE \mathcal{R} , for a linear duration invariant \mathcal{D} , if we can effectively find a normal form \mathcal{N} such that $\mathcal{R} \models \mathcal{D}$ if and only if $\mathcal{N} \models \mathcal{D}$, then we can check $\mathcal{R} \models \mathcal{D}$ effectively.

for an HRE \mathcal{R} and a linear duration invariant \mathcal{D} , we attempt to find a normal form \mathcal{N} such that $\mathcal{L}(\mathcal{R}) \models \mathcal{D}$ if and only if $\mathcal{L}(\mathcal{N}) \models \mathcal{D}$ by the following procedure:

Step 0. Let $\mathcal{R}' := \mathcal{R}$.

Step 1. For \mathcal{R}' , distributing \wedge over \oplus , and $[a, b]$ over \oplus , we obtain \mathcal{Q} . If \mathcal{Q} is a normal form, then we are done.

Step 2. For a sub-expression \mathcal{Q}_S of \mathcal{Q} which is of the form $\mathcal{Q}_S = \mathcal{Q}_1^*$, replacing an occurrence of \mathcal{Q}_S in \mathcal{Q} with X , we obtain a context $\mathcal{C}_\mathcal{Q}(X)$ such that $\mathcal{R}' = \mathcal{C}_\mathcal{Q}(\mathcal{Q}_S)$.

Step 3. Finding an HRE \mathcal{Q}'_S in which there is no occurrence of combinator $*$ such that $\mathcal{C}_\mathcal{Q}(\mathcal{Q}_S) \models \mathcal{D}$ if and only if $\mathcal{C}_\mathcal{Q}(\mathcal{Q}'_S) \models \mathcal{D}$. Let $\mathcal{R}' := \mathcal{C}_\mathcal{Q}(\mathcal{Q}'_S)$, and go to Step 1. \square

Obviously the procedure is correct. The problem is how to find \mathcal{Q}'_S in Step 3. The following lemmas and theorems will help to solve that problem.

Let $\mathcal{C}(X)$ be a context.

Lemma 4. Let \mathcal{R} and \mathcal{R}' be HREs. If for any $\sigma \in \mathcal{L}(\mathcal{R})$, there is $\sigma' \in \mathcal{L}(\mathcal{R}')$ such that $\tau(\sigma) = \tau(\sigma')$ and $\theta(\sigma, \mathcal{D}) \leq \theta(\sigma', \mathcal{D})$, then $\mathcal{C}(\mathcal{R}') \models \mathcal{D}$ implies $\mathcal{C}(\mathcal{R}) \models \mathcal{D}$. \square

Lemma 5. Suppose $\omega(\mathcal{C}(X)) = \infty$, and \mathcal{R} be a nonzero-simple HRE \mathcal{R} such that $M_\theta(\mathcal{R}) \leq 0$. Then for any real number N_t , for any $\sigma \in \mathcal{L}(\mathcal{C}(\mathcal{R}^*))$ such that $\tau(\sigma) \geq N_t$, there is $\sigma' \in \mathcal{L}(\mathcal{C}(\oplus_{j=0}^p \mathcal{R}^j))$ such that

$$\tau(\sigma') \geq N_t \text{ and } \theta(\sigma, \mathcal{D}) \leq \theta(\sigma', \mathcal{D}),$$

where $p = (\lfloor h/m_\tau(\mathcal{R}) \rfloor + 1)$, and $h = \max(\varphi(\mathcal{C}(X), N_t)$. \square

Lemma 6. Suppose $\omega(\mathcal{C}(X)) = \infty$, and \mathcal{R} be a nonzero-simple HRE such that $M_\theta(\mathcal{R}) > 0$. Then for any nonnegative real numbers N_t and M_r , there is $\sigma \in \mathcal{L}(\mathcal{C}(\mathcal{R}^*))$ such that $\tau(\sigma) \geq N_t$ and $\theta(\sigma, \mathcal{D}) > M_r$. \square

Lemma 7. Suppose \mathcal{R} be a nonzero-simple HRE, and N_t be a nonnegative real number. Then for any $\sigma \in \mathcal{L}(\mathcal{C}(\mathcal{R}^*))$, $\tau(\sigma) \leq N_t$ implies $\sigma \in \mathcal{L}(\mathcal{C}(\oplus_{j=0}^p \mathcal{R}^j))$, where $p = \lfloor N_t/m_\tau(\mathcal{R}) \rfloor + 1$. \square

Lemma 8. Let \mathcal{R} be a nonzero-simple HRE. Then $\mathcal{L}(\mathcal{C}(\oplus_{j=0}^p \mathcal{R}^j)) \supseteq \mathcal{L}(\mathcal{C}(\mathcal{R}^*))$, where $p = \lfloor \omega(\mathcal{C}(X))/m_\tau(\mathcal{R}) \rfloor + 1$. \square

These lemmas can be proved by induction on the structure of context. Their detailed proofs are omitted because of space consideration. From these lemmas, we can prove the following theorems.

Theorem 7. Let \mathcal{R}_1 and \mathcal{R}_2 be HREs. Then

$$\mathcal{C}((\mathcal{R}_1 \oplus \mathcal{R}_2)^*) \models \mathcal{D} \text{ iff } \mathcal{C}((\mathcal{R}_1^*)^\wedge (\mathcal{R}_2^*)) \models \mathcal{D}.$$

Proof. By Definition 1, $\mathcal{L}((\mathcal{R}_1^*)^\wedge (\mathcal{R}_2^*)) \subseteq \mathcal{L}((\mathcal{R}_1 \oplus \mathcal{R}_2)^*)$. From Lemma 4, the half of the claim follows, i.e. $\mathcal{C}((\mathcal{R}_1 \oplus \mathcal{R}_2)^*) \models \mathcal{D}$ implies $\mathcal{C}((\mathcal{R}_1^*)^\wedge (\mathcal{R}_2^*)) \models \mathcal{D}$. The other half can be proved as follows. For any $\sigma_1 \in \mathcal{L}(\mathcal{R}_1)$ and $\sigma_2 \in \mathcal{L}(\mathcal{R}_2)$, since $\tau(\sigma_1 \hat{\sigma}_2) = \tau(\sigma_1) + \tau(\sigma_2)$ and $\theta(\sigma_1 \hat{\sigma}_2, \mathcal{D}) = \theta(\sigma_1, \mathcal{D}) + \theta(\sigma_2, \mathcal{D})$, we have $\tau(\sigma_1 \hat{\sigma}_2) = \tau(\sigma_2 \hat{\sigma}_1)$ and $\theta(\sigma_1 \hat{\sigma}_2, \mathcal{D}) = \theta(\sigma_2 \hat{\sigma}_1, \mathcal{D})$. Therefore, any $\sigma \in \mathcal{L}(\mathcal{C}((\mathcal{R}_1 \oplus \mathcal{R}_2)^*))$ can be permuted into $\sigma' \in \mathcal{L}(\mathcal{C}((\mathcal{R}_1^*)^\wedge (\mathcal{R}_2^*)))$. Hence, from Lemma 4, the result follows. \square

Theorem 8. Let $\mathcal{R} = (\langle v_1, \lambda_1 \rangle^\wedge \langle v_2, \lambda_2 \rangle^\wedge \dots^\wedge \langle v_m, \lambda \rangle, \Delta)$ be a zero-simple HRE. Let Δ' be the set

$$\{0 \leq \sum_{i=1}^m c_i \lambda_i \mid 0 \leq \sum_{i=1}^m c_i \lambda_i \leq b \in \Delta \wedge \exists j \cdot (1 \leq j \leq m \wedge c_j < 0)\},$$

and $\mathcal{R}' = (\langle v_1, \lambda_1 \rangle^\wedge \langle v_2, \lambda_2 \rangle^\wedge \dots^\wedge \langle v_m, \lambda \rangle, \Delta')$. Then $\mathcal{C}(\mathcal{R}^*) \models \mathcal{D}$ iff $\mathcal{C}(\mathcal{R}') \models \mathcal{D}$.

Proof. Before the proof, we should note that by the definition of zero-simple HREs, $\tau(\mathcal{R}) = 0$ implies that for any inequality $a \leq c_1\lambda_1 + c_2\lambda_2 + \dots + c_m\lambda_m \leq b$ in Δ , $a \leq 0$ and $b \geq 0$.

The half of the claim that $\mathcal{C}(\mathcal{R}') \models \mathcal{D}$ implies $\mathcal{C}(\mathcal{R}^*) \models \mathcal{D}$, is explained as follows. By Definition 1, any $\sigma \in \mathcal{L}(\mathcal{R}^*)$ is of the form $\sigma_1 \hat{\ } \sigma_2 \hat{\ } \dots \hat{\ } \sigma_n$, where

$$\sigma_i = (v_1, t_{i1}) \hat{\ } (v_2, t_{i2}) \hat{\ } \dots \hat{\ } (v_m, t_{im}) \in \mathcal{L}(\mathcal{A}) \quad (i = 1, 2, \dots, n).$$

For any j ($1 \leq j \leq m$), let $t'_j = t_{1j} + t_{2j} + \dots + t_{nj}$, and let

$$\sigma' = (v_1, t'_1) \hat{\ } (v_2, t'_2) \hat{\ } \dots \hat{\ } (v_m, t'_m).$$

Since for any i ($1 \leq i \leq n$), $t_{i1}, t_{i2}, \dots, t_{im}$ satisfy Δ , t'_1, t'_2, \dots, t'_m satisfy Δ' as well. It follows that $\sigma' \in \mathcal{L}(\mathcal{R}')$. Since $\theta(\sigma, \mathcal{D}) = \theta(\sigma', \mathcal{D})$ and $\tau(\sigma) = \tau(\sigma')$, the first half of the claim follows from Lemma 4.

The other half of the claim, i.e. $\mathcal{C}(\mathcal{R}^*) \models \mathcal{D}$ implies $\mathcal{C}(\mathcal{R}') \models \mathcal{D}$, can be proved as follows. For any $\sigma' = (v_1, t'_1) \hat{\ } (v_2, t'_2) \hat{\ } \dots \hat{\ } (v_m, t'_m) \in \mathcal{L}(\mathcal{R}')$, since t_1, t_2, \dots, t_m satisfy Δ' , for any $0 \leq \sum_{i=1}^m c_i \lambda_i \leq b \in \Delta$, we have $\sum_{i=1}^m c_i t_i \geq 0$. Because for each inequality $a \leq c_1\lambda_1 + c_2\lambda_2 + \dots + c_m\lambda_m \leq b$ in Δ , $a \leq 0$ and $b \geq 0$, and because Δ is a finite set, we can choose a natural number p such that for any inequality $a \leq c_1\lambda_1 + c_2\lambda_2 + \dots + c_m\lambda_m \leq b \in \Delta$,

$$a \leq \frac{c_1 t_1 + c_2 t_2 + \dots + c_m t_m}{p} \leq b.$$

For each i ($1 \leq i \leq m$), let $b_i = t_i/p$, and let $\sigma_b = (v_1, b_1) \hat{\ } (v_2, b_2) \hat{\ } \dots \hat{\ } (v_m, b_m)$. Obviously, $\sigma \in \mathcal{L}(\mathcal{R})$. Let

$$\sigma = \underbrace{\sigma_b \hat{\ } \sigma_b \hat{\ } \dots \hat{\ } \sigma_b}_p.$$

It follows that $\sigma \in \mathcal{L}(\mathcal{R}^*)$. Since $\theta(\sigma, \mathcal{D}) = \theta(\sigma', \mathcal{D})$ and $\tau(\sigma) = \tau(\sigma')$, by Lemma 4, $\mathcal{C}(\mathcal{R}^*) \models \mathcal{D}$ implies $\mathcal{C}(\mathcal{R}') \models \mathcal{D}$. \square

Theorem 9. Suppose $\omega(\mathcal{C}(X)) = \infty$, $T = \infty$, and \mathcal{R} be a nonzero-simple HRE such that $M_\theta(\mathcal{R}) > 0$. Then $\mathcal{C}(\mathcal{R}^*) \not\models \mathcal{D}$.

Proof. The theorem follows immediately from Lemma 6. \square

Theorem 10. Suppose $\omega(\mathcal{C}(X)) = \infty$, $T = \infty$, and \mathcal{R} be a nonzero-simple HRE such that $M_\theta(\mathcal{R}) \leq 0$. Then $\mathcal{C}(\mathcal{R}^*) \models \mathcal{D}$ iff $\mathcal{C}(\oplus_{j=0}^p \mathcal{R}^j) \models \mathcal{D}$, where $p = (\lfloor h/m_\tau(\mathcal{R}) \rfloor + 1)$, $h = \max(\varphi(\mathcal{C}(X), t)$.

Proof. By Definition 1, $\mathcal{L}(\mathcal{R}^*) \supseteq \mathcal{L}(\oplus_{j=0}^p \mathcal{R}^j)$ holds, which by Lemma 4 implies a half of the claim, i.e. $\mathcal{C}(\mathcal{R}^*) \models \mathcal{D}$ implies $\mathcal{C}(\oplus_{j=0}^p \mathcal{R}^j) \models \mathcal{D}$. The other half is straightforward from Lemma 5. \square

Theorem 11. Suppose $\omega(\mathcal{C}(X)) \neq \infty$ or $T \neq \infty$, and \mathcal{R} be a nonzero-simple HRE. Then $\mathcal{C}(\mathcal{R}^*) \models \mathcal{D}$ iff $\mathcal{C}(\oplus_{j=0}^p \mathcal{R}^j) \models \mathcal{D}$, where $p = (\lfloor h/m_\tau(\mathcal{R}) \rfloor + 1)$, $h = \min(\omega(\mathcal{C}(X), T)$.

Proof. One half of the claim, i.e. $\mathcal{C}(\mathcal{R}^*) \models \mathcal{D}$ implies $\mathcal{C}(\oplus_{j=0}^p \mathcal{R}^j) \models \mathcal{D}$ is exactly the same as the proof of Theorem 10. The other half of the claim is a direct consequence of Lemmas 7 and 8. \square

Based on the above theorems, the algorithm to check an HRE \mathcal{R} for a linear duration invariant \mathcal{D} is now described as follows.

Step 0. Let $\mathcal{R}' := \mathcal{R}$.

Step 1. For \mathcal{R}' , distributing \wedge over \oplus , and $[a, b]$ over \oplus , we obtain \mathcal{Q} .

Step 2. Finding a sub-expression \mathcal{Q}_S of \mathcal{Q} which has one of the following three forms:

1. $\mathcal{Q}_S = (\mathcal{R}_1 \oplus \mathcal{R}_2 \oplus \dots \oplus \mathcal{R}_k)^*$ ($k \geq 2$), where every \mathcal{R}_i ($1 \leq i \leq m$) is a simple HRE.
2. $\mathcal{Q}_S = \mathcal{R}_1^*$, where \mathcal{R}_1 is a nonzero-simple HRE.
3. $\mathcal{Q}_S = \mathcal{R}_1^*$, where \mathcal{R}_1 is a zero-simple HRE.

If such \mathcal{Q}_S could not be found, goto Step 6 (note that it is not difficult to prove that if we can not find out such a \mathcal{Q}_S , then \mathcal{Q} is a normal form); otherwise replacing the occurrence of \mathcal{Q}_S in \mathcal{Q} with X , we get a context $\mathcal{C}_Q(X)$ such that $\mathcal{Q} = \mathcal{C}_Q(\mathcal{Q}_S)$. Then, if \mathcal{Q}_S has the first form, goto Step 3; if \mathcal{Q}_S has second form, goto Step 4; if \mathcal{Q}_S has the third form, goto Step 5.

Step 3. By Theorem 7, we transform \mathcal{Q} into $\mathcal{Q}' = \mathcal{C}_Q((\mathcal{R}_1)^* \wedge (\mathcal{R}_2)^* \wedge \dots \wedge (\mathcal{R}_m)^*)$. Thus, let $\mathcal{R}' := \mathcal{Q}'$, and goto Step 1.

Step 4. We first calculate $\omega(\mathcal{C}_Q(X))$ and $M_\theta(\mathcal{R}_1)$. If $\omega(\mathcal{C}_Q(X)) \neq \infty$ or $T \neq \infty$, then by Theorem 11, we transform \mathcal{Q} into $\mathcal{Q}' = \mathcal{C}_Q(\oplus_{j=0}^p \mathcal{R}_1^j)$, where $p = (\lfloor h/m_\tau(\mathcal{R}_1) \rfloor + 1)$, and $h = \min(\omega(\mathcal{C}_Q(X)), T)$. Therefore, let $\mathcal{R}' := \mathcal{Q}'$, and goto Step 1.

Otherwise, $\omega(\mathcal{C}_Q(X)) = \infty$ and $T = \infty$. If $M_\theta(\mathcal{R}_1) > 0$, then by Theorem 9, we conclude $\mathcal{C}_Q(\mathcal{R}_1^*) \not\models \mathcal{D}$ and exit. Otherwise, by Theorem 10, we transform \mathcal{Q} into $\mathcal{Q}' = \mathcal{C}_Q(\oplus_{j=0}^p \mathcal{R}_1^j)$, where $p = (\lfloor h/m_\tau(\mathcal{R}_1) \rfloor + 1)$, and $h = \max(\varphi(\mathcal{C}_Q(X), t)$. Let $\mathcal{R}' := \mathcal{Q}'$, and goto Step 1.

Step 5. By Theorem 8, we transform \mathcal{Q} into $\mathcal{Q}' = \mathcal{C}_Q(\mathcal{R}_1')$, where \mathcal{R}_1' is the simple HRE defined from \mathcal{R}_1 in Theorem 2. Let $\mathcal{R}' := \mathcal{Q}'$, and goto Step 1.

Step 6. Since \mathcal{Q} is a normal form now, we check $\mathcal{Q} \models \mathcal{D}$ by linear programming. If $\mathcal{Q} \models \mathcal{D}$, then $\mathcal{R} \models \mathcal{D}$; otherwise $\mathcal{R} \not\models \mathcal{D}$. \square

5 Conclusion

In this paper, we introduce Hybrid Regular Expression to define a class of linear hybrid automata for which two class of reachability problems and the satisfaction problem for linear duration invariants are decidable. We use linear programming techniques for checking this class of linear hybrid automata. The idea comes from [4] in which the satisfaction problem of linear duration invariants for a simple class of real-time automata is solved by linear programming techniques, which is well established. In [5] the problem for timed automata has been solved by

mixed integer/linear programming techniques. Because of the advantages of the approach of [4] in comparison to the others, in [9] we have generalised it to a subclass of timed automata. In [10], by developing the techniques in [4,9], we show that by linear programming technique the problem can be solved totally for a class of linear hybrid automata. In this paper, we use similar techniques to define a larger decidable class of linear hybrid automata which includes the class of linear hybrid automata defined in [10].

We note the work in [8] in which *timed regular expression* of the same expressive power as timed automata is introduced. We are inspired by it and attempt to extend Hybrid Regular Expression such that it has the same expressive power as linear hybrid automata in the future.

References

1. Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS 1996)*, pp. 278-292.
2. R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, S. Yovine. The algorithmic analysis of hybrid systems. In *Theoretical Computer Science*, 138(1995), pp.3-34.
3. Zhou Chaochen, C.A.R. Hoare, A.P. Ravn. A Calculus of Durations. In *Information Processing Letter*, 40, 5, 1991, pp.269-276.
4. Zhou Chaochen, Zhang Jingzhong, Yang Lu and Li Xiaoshan. Linear Duration Invariants. In *Formal Techniques in Real-Time and Fault-Tolerant Systems, LNCS 863*, pp.88-109.
5. Y. Kesten, A. Pnueli, J. Sifakis, S. Yovine. Integration Graphs: A Class of Decidable Hybrid Systems. In *Hybrid System, LNCS 736*, pp.179-208.
6. Rajeev Alur, Costas Courcoudetis, and Thomas A. Henzinger. Computing Accumulated Delays in Real-time Systems. In *Proc. CAV'93, LNCS 818*, pp.181-193.
7. Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's Decidable About Hybrid Automata? In *Proc. of the 27th Annual ACM Symposium on Theory of Computing (STOC)*, 1995, pp.373-382.
8. Eugene Asarin, Paul Caspi, Oded Maler. A Kleene Theorem for Timed Automata. In *Proceedings of Logic in Computer Science*, IEEE Comp. Soc., 1997.
9. Li Xuandong, Dang Van Hung. Checking Linear Duration Invariants by Linear Programming. In *Concurrence and Parallelism, Programming, Networking, and Security, LNCS 1179*, pp.321-332.
10. Li Xuandong, Dang Van Hung, and Zheng Tao. Checking Hybrid Automata for Linear Duration Invariants. In *Advance in Computing Science - ASIAN'97, LNCS 1345*, pp.166-180.

Stabilization of Systems with Changing Dynamics

Miloš Žefran and Joel W. Burdick

Department of Mechanical Engineering, MC 104-44
California Institute of Technology, Pasadena, CA 91125

Abstract. We present a framework for designing stable control schemes for systems whose dynamic equations change as they evolve on the state space. It is usually difficult or even impossible to design a single controller that would stabilize such a system. An appealing alternative are switching control schemes, where a different controller is employed on each of the regions defined by different dynamic characteristics and the stability of the overall system is ensured through appropriate switching scheme. We derive sufficient conditions for the stability of a switching control scheme in a form that can be used for controller design. An important feature of the proposed framework is that although the overall hierarchy can be very complicated, the stability depends only on the immediate relation of each controller to its neighbors. This makes the application of our results particularly straight forward. The methodology is applied to stabilization of a shimmying wheel, where changes in the dynamics are due to switches between sliding and rolling.

1 Introduction

The design of controllers for hybrid systems is a difficult problem that is still not satisfactorily solved. Most existing design methodologies assume that the underlying dynamics are continuous and that the hybrid behavior arises because the system must perform several functions. The control synthesis task is then to design controllers that achieve each of the functions and a coordination scheme that guarantees that properties like safety and liveness are satisfied at all times. This work addresses a different problem. We study dynamical systems that change their dynamic behavior as they evolve in the state space. The hybrid nature is thus inherent in the dynamics of the system and does not come from the controller specification. In this paper we study the problem of stabilization of such systems. The goal is to design a controller that stabilizes an equilibrium set in one of the regions, moving through other regions if necessary. We achieve this by designing a controller on each of the regions and a scheme for switching between these controllers. We show that the stability of the overall system can be guaranteed by imposing conditions on controllers that operate on adjacent regions. This leads to modularity of the design process and considerably simplifies the synthesis problem. The stability analysis is based on Lyapunov functions.

A starting point for controller design is a choice of a formalism for description of a hybrid system. In the literature we can find several alternatives. Alur et al. [1] and Nicollin et al. [2] defined the notion of hybrid automaton, building their

work on the automata theory. Brockett [3] devised his model using the theory of dynamical systems. Other works in this category are [4] and [5]. Branicky gives an overview of such models and relates them to his own model [6]. We use models in this second group for our work.

Prior work on hybrid controller design has often been limited to specific applications. Lygeros et al. [7] proposed a game-theoretic framework for design of controllers for intelligent highway systems and air traffic control systems. Puri [8] and Deshpande [9] developed methods for controller design using a simplified version of hybrid automata. Kohn et al. developed a methodology for coordination of multiple agents [10]. Branicky & Mitter [11] and Žefran et al. [12] employed optimal control for synthesis of open-loop trajectories. Goodwine & Burdick [13] developed a controllability test and a planning method for a class of hybrid systems called stratified systems. An important step in controller design is verification. The approaches in [7]–[9] include verification as an integral part of the design process. Some other works that address the verification are [14], [15], and [16].

A number of authors considered stability of hybrid controllers. Branicky [17] devised sufficient conditions for stability of a system that switches between different controllers that stabilize an equilibrium point. Based on this work, Malmberg et al. [18] proposed a strategy for choosing a controller among several available controllers so that the overall system is stable. Both papers allow dynamic equations to change, but they are primarily concerned with the case when the equilibrium point is the same for each controller so there is no need to actively drive the system into some designated region, as we do in the present paper. An earlier work on stability of switching controllers is also [19].

The idea of driving the system through a sequence of equilibrium points until a desired equilibrium point is reached was employed in [20]. In this work, the switch between different controllers always occurs at an equilibrium point. The authors also assume that the region of attraction of each controller is known so there is no need for Lyapunov functions to prove the stability.

The paper is organized as follows. We start with a motivating example and introduce some notions for stability analysis on manifolds. We next formulate three propositions that give sufficient conditions for the stability of a switching controller. The propositions are progressively less abstract and lead to a practical synthesis methodology. We then apply the methodology to solve the problem of stabilization for the classical shimmying wheel. We demonstrate the behavior of the controller with some simulation results and conclude the paper with a brief discussion.

2 Preliminaries

To motivate the theoretical development we start with an example. The system that we study is the classical shimmying wheel [21, 22]. A schematic of the shimmying wheel is shown in Fig. 1. A rigid link with a wheel is attached to a hinge joint, which is in turn connected to a rigid object through a sliding joint

between two springs (Fig. 1). The control input is the torque at the hinge joint. The object moves with a constant velocity v in the direction perpendicular to the axis of the sliding joint. The shimmying wheel can be seen as a simplified model of an aircraft nose wheel or a motorcycle front wheel, with the springs modeling the compliance of the wheel and the wheel attachment [22]. It can also serve as a model of a vehicle towing a trailer, with the springs abstracting the compliance in the kingpin.

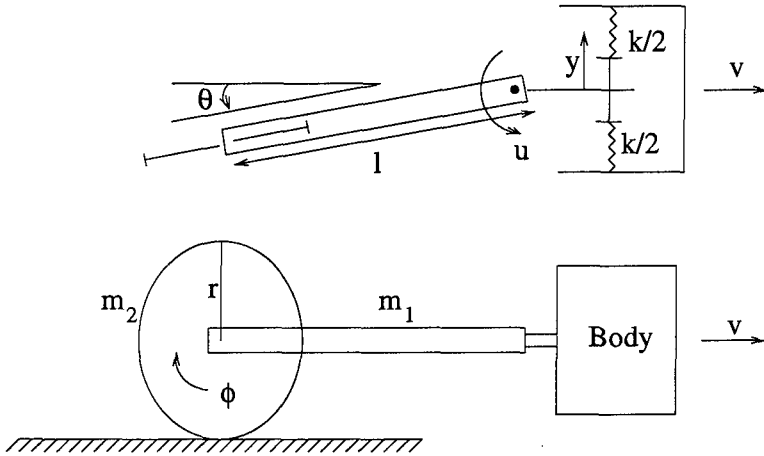


Fig. 1. A top view and a side view of a shimmying wheel.

The goal of control is to stabilize the wheel so that the bar is aligned with the direction of v (perpendicular to the sliding axis) and the slider is in the neutral position between the two springs (the forces of the springs are equal in magnitude and of the opposite sign). This task is complicated by the fact that the system can operate in two regimes: the wheel can either roll without sliding or it can slip. The slipping regime is undesirable, but often unavoidable. The system will switch between rolling and sliding depending on the magnitude of the contact force between the wheel and the ground: the wheel will slip if the force in rolling would be greater than the friction force. If we assume a feedback control law for the torque about the hinge joint, the contact force is completely determined by the state of the system and the state space gets divided into two regions separated by a switching surface on which the contact force equals the friction force. In each of the regions the equations of motion are different. It is therefore unlikely that a single controller could stabilize the system and even if one exists it is not clear how to design it.

A controller that is designed without taking the hybrid nature of the dynamics into account can produce undesired results. It is for example possible to design a stable controller that linearizes the shimmying wheel dynamics if the wheel is rolling. Figure 2(a) shows that this controller efficiently stabilizes the

system. However, if the same controller is used while the wheel is sliding, *it can destabilize the system*, as can be seen in Fig. 2(b). This example shows that a more comprehensive approach to design of controllers for systems with hybrid dynamics is needed.

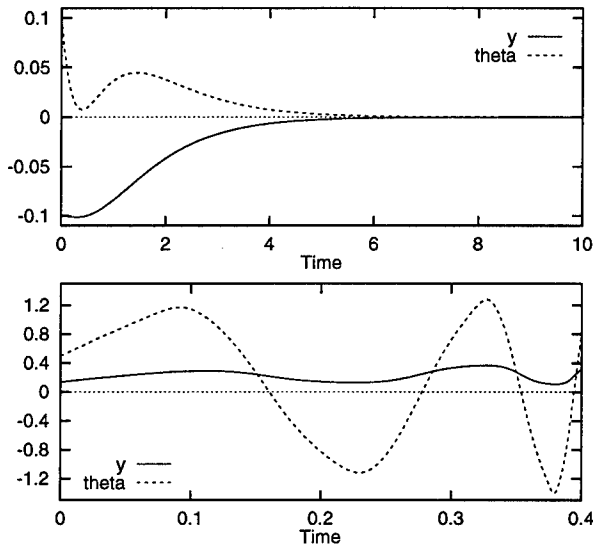


Fig. 2. A linearizing controller applied in rolling (a) and sliding (b).

2.1 Stability theory on manifolds

We are interested in stabilizing submanifolds (possibly unbounded). Conventional Lyapunov theory can not be directly applied to this setting, so we need to introduce some additional concepts (see [23]).

Definition 1. A *distance* between a point x and a set $E \subseteq \mathbb{R}^n$ is defined by:

$$\rho(x, E) = \inf_{y \in E} d(x, y) \quad (1)$$

A *ball* with radius R around E is the set $B(E, R) = \{x \mid \rho(x, E) < R\}$.

Definition 2. A smooth manifold $E \subset M$ is *locally stable* if for any $R > 0$ there exist $r > 0$ such that if $\rho(x(t_0), E) < r$ then $\rho(x(t), E) < R$ for every $t > t_0$. If, in addition, $\lim_{t \rightarrow \infty} \rho(x(t), E) = 0$, then we say that E is *locally asymptotically stable*.

Definition 3. A submanifold $E \subset M$ is *locally attractive* if there exists $R > 0$ such that if $\rho(x(t_0), E) < R$ then $\lim_{t \rightarrow \infty} \rho(x(t), E) = 0$. We also say that trajectories starting inside $B(E, R)$ converge to E .

Theorem 4 [24, 25]. *If for a control system Σ there exists a C^1 function $V : M \rightarrow \mathbb{R}$, such that:*

- (1) $V(x) \geq 0$ and $V(x) = 0 \Leftrightarrow x \in E$;
- (2) *there exists a monotonically increasing function $\alpha : \mathbb{R}^+ \rightarrow \mathbb{R}^+$, $\alpha(0) = 0$, such that $\alpha(\rho(x, E)) < V(x)$;*
- (3) *there exists a monotonically increasing function $\beta : \mathbb{R}^+ \rightarrow \mathbb{R}^+$, $\beta(0) = 0$, such that $V(x) < \beta(\rho(x, E))$;*
- (4) $\dot{V}(x) \leq 0$, where \dot{V} is the derivative of V along the trajectories of Σ ;

then the manifold E is locally stable. If in addition:

- (5) *there exists a monotonically increasing function $\gamma : \mathbb{R}^+ \rightarrow \mathbb{R}^+$, $\gamma(0) = 0$, such that $\dot{V}(x) \leq -\gamma(\rho(x, E)) < 0$,*

then E is locally asymptotically stable.

2.2 Modeling

In this section we describe the setting which will be used to formally describe systems whose dynamics change. Suppose we have a dynamical system Σ and a collection of (differentiable, connected) manifolds $\mathcal{M} = \{M_1, M_2, \dots, M_n\}$. The manifolds need not be disjoint, they can be a subset of each other and in some cases it will be even convenient to take some of them to be equal. This collection of manifolds must reflect the changing dynamics, but additional manifolds can be defined for the purposes of a particular application. An example of a collection of manifolds is shown in Fig. 3(a). On each manifold, the system is described with a set of equations:

$$\dot{x}_i = f_i(x_i, u_i, t), \quad (2)$$

where x_i is the state of the system and u_i is the vector of inputs for the system evolving on the submanifold M_i . In general, f_i 's can be different to reflect changes in the dynamics of the system. Also the dimensions of the manifolds might be different. For example, in the case of the shimmying wheel, the manifolds M_1 and M_2 would correspond to sliding and rolling, respectively, where M_1 is the whole space and M_2 is the subspace on which the rolling constraint is satisfied.

We will assume that on each manifold M_i we design a controller g_i :

$$u_i = g_i(x_i, t), \quad (3)$$

The reason of allowing some manifolds in the collection \mathcal{M} to be the same is that we may wish to define different controllers on the same physical space. Let $E_n \subseteq M_n$ be a manifold to which we wish to steer the system Σ . The problem that we address in this paper is how to design the controllers g_i and a rule for

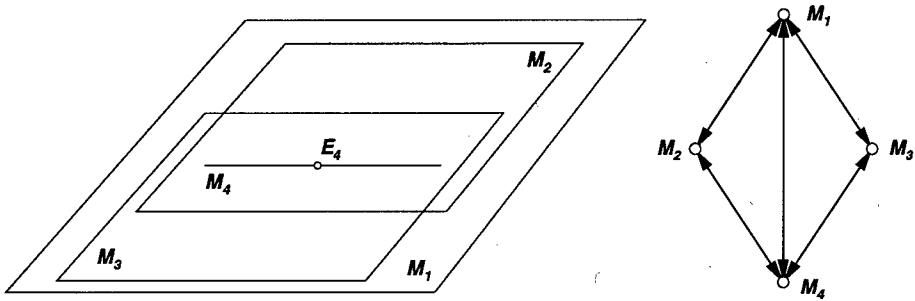


Fig. 3. (a) A sequence of embedded manifolds; (b) the corresponding graph.

switching among them (a switching scheme) that stabilizes the system to E_n (if possible globally). This task is complicated by the fact that, in general, we do not know the sequence of the manifolds that the dynamical system will traverse. Take for example the shimmying wheel. If the system is rolling, the controller action might cause the wheel to slip, but it is conceivable that within a certain region such switching does not happen. And it is of course always possible that a disturbance (for example a slippery patch) causes the rolling wheel to slip.

The topology of a system evolving on a collection of manifolds can be described with a graph. The vertices of the graph correspond to different manifolds. There will be an edge from a manifold M_i to a manifold M_j if it is possible to switch from M_i to M_j (there exists a trajectory that passes from M_i to M_j). For example, if we assume that a nonempty intersection of two manifolds implies that it is possible to pass between the manifolds, the graph for the system in Fig. 3(a) would be Fig. 3(b).

3 Sufficient conditions for stability

Take a control system Σ evolving on the collection of manifolds \mathcal{M} . Assume that on each manifold M_i , we have a controller g_i (i.e., $u_i = g_i(x, t)$). Let the controller g_n stabilize the manifold E_n (i.e., the target manifold). Assume we can construct a Lyapunov function V_n which satisfies the conditions (1)-(5) of Theorem 4. Let

$$\begin{aligned} \mathcal{S}: \quad \mathbb{R}^n \times \{1, \dots, n\} &\rightarrow \{1, \dots, n\} \\ (x, \eta) &\mapsto \mathcal{S}(x, \eta) \end{aligned} \quad (4)$$

denote the switching scheme. In other words, the function \mathcal{S} selects the controller to be used, depending on the state x , and the controller that is currently used, η . Clearly, $\mathcal{S}(x, \eta) = i$ implies $x \in M_i$, since g_i is only defined on M_i . The following proposition gives sufficient conditions for E_n to be globally attractive:

Proposition 5. *Let the switching scheme \mathcal{S} satisfy the following conditions:*

1. There exists $L > 0$ such that $S(x, n) = n$ for every $x \in B(E_n, L) \cap M_n$.
2. For any trajectory $x(t)$ there exists a $\Delta > 0$ and an infinite sequence $\{t_i\}$ whose elements satisfy:
 - (a) for every $t \in [t_i, t_i + \Delta]$, $S(x(t), \eta(t)) = n$;
 - (b) $V(t_i + \Delta) \geq V(t_{i+1})$.

Then the submanifold E_n is globally attractive.

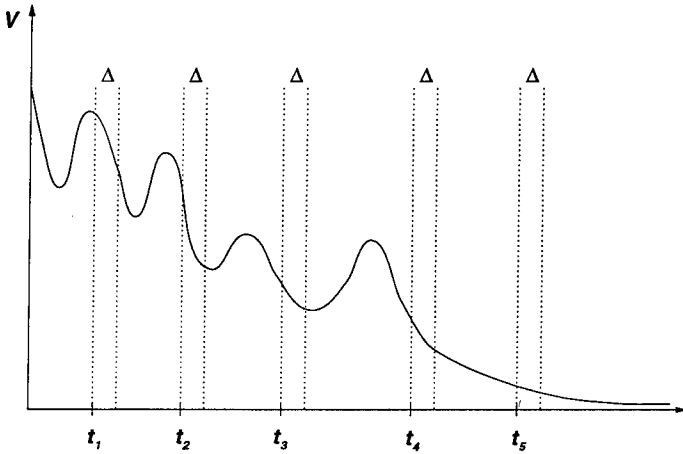


Fig. 4. Values of the Lyapunov function and a sequence satisfying condition 2(b) of Proposition 5.

Remark: Condition (1) guarantees that there is a region around E_n in which it is not possible to switch from g_n to some other controller g_i . That is, we assume that the controller g_n can capture and stabilize Σ in some region around E_n . Condition (2) states that regardless of the current state, the system will eventually come under the control of g_n and stay under the control of g_n for at least time Δ . Furthermore, we can find a sequence of time subintervals of length at least Δ so that the Lyapunov function restricted to the union of these intervals is monotonically decreasing.

Proof. Let $\{t_k\}$ be a sequence given by condition (2). Since the Lyapunov function V is monotonically decreasing when the system evolves on M_n , condition 2(b) implies that $t_{i+1} - t_i \geq \Delta$. Now take $I_n = \cup_{k \in \mathbb{N}} [t_k, t_k + \Delta]$ and consider the system evolving on I_n . By assumption, V satisfies the conditions of Theorem 4, so we can find monotonically increasing functions α , β and γ such that $\alpha(\rho(x, E_n)) < V(x) < \beta(\rho(x, E_n))$ and $\dot{V}(x) \leq -\gamma(\rho(x, E_n)) < 0$. Let $r = \rho(x(t_0), E_n)$ and let ϵ be an arbitrary number such that $0 < \epsilon < r$. Then we can find $\delta > 0$ such that $\beta(\delta) < \alpha(\epsilon)$. Let K be an integer such that $K > \frac{\beta(r)}{\Delta\gamma(\delta)}$

and take $\tau = t_K + \Delta$. Suppose that $\rho(x(t), E_n) > \epsilon$ for every $t \in I_n \cap [t_0, \tau]$. Then we have:

$$\begin{aligned}
 0 < \alpha(\epsilon) &\leq V(x(\tau)) = V(x(t_K)) + \int_{t_K}^{t_K+\Delta} \dot{V}(x(t)) dt \\
 &\leq V(x(t_K)) - \int_{t_K}^{t_K+\Delta} \gamma(\rho(x(t), E_n)) dt \leq V(x(t_K)) - \int_{t_K}^{t_K+\Delta} \gamma(\delta) dt \\
 &= V(x(t_K)) - \Delta\gamma(\delta) \leq V(x(t_{K-1})) - \Delta\gamma(\delta) \leq \dots \\
 &\leq V(x(t_0)) - K\Delta\gamma(\delta) \leq \beta(r) - K\Delta\gamma(\delta) < 0
 \end{aligned} \tag{5}$$

This is a contradiction, implying that there exists $\tau' \in I_n \cap [t_0, \tau]$ such that $\rho(x(\tau'), E_n) < \delta$. But then for every $t \in I_n$ such that $t > \tau'$:

$$\alpha(\rho(x(t), E_n)) \leq V(x(t)) \leq V(x(\tau')) \leq \beta(\delta) < \alpha(\epsilon)$$

which implies:

$$\rho(x(t), E_n) < \epsilon \quad \forall t > \tau', t \in I_n$$

This shows that $\rho(x(t), E_n)$ converges to 0 on I_n .

Since $\rho(x(t), E_n)$ converges to 0 on I_n , there exists $T > 0$ such that for all $t > T, t \in I_n$, $\rho(x(t), E_n) < L$. But by assumption, for $x \in B(E_n, L) \cap M_n$ the system can not switch from M_n to some $M_j, j \neq i$, which means that the system will stay under the control of g_n for all $t > T$ and therefore converge to E_n .

While the lemma provides sufficient conditions for convergence of the system trajectories to E_n , these conditions are difficult to check and therefore not suitable for controller design. It is particularly difficult to check condition (2). We therefore provide two additional tests that are less general, but are easier to apply.

Take M_1, M_2, \dots, M_n , the collection of manifolds on which a dynamical system evolves, and let $A = \{1, 2, \dots, n\}$ be the index set. The switching scheme S defines a relation $\text{Switch}(A)$, if we put $\text{Switch}(i, j)$ when it is possible to switch from the manifold M_i (controller g_i) to the manifold M_j (controller g_j). More formally:

$$\text{Switch}(A) = \{(i, j) \mid \exists x \in M_i \text{ s.t. } S(x, i) = j\} \tag{6}$$

Note that the graph representing this relation is precisely the graph described in Section 2.2. We can then show:

Proposition 6. *Let \leq be a partial order within the transitive closure of the relation $\text{Switch}(A)$ which has the smallest element, and let this smallest element be n . Assume that the switching scheme S has the following properties:*

1. *There exists $L > 0$ such that $S(x, n) = n$ for every $x \in B(E_n, L) \cap M_n$.*
2. *If $x(t)$ is a trajectory of Σ and $M_i, i \neq n$ is a manifold on which $x(t)$ evolves for an infinite amount of time, then there exists $\Delta > 0$ such that for every T we can find $\tau > T$ such that $S(x(t), \eta(t)) < i$ for every $t \in [\tau, \tau + \Delta]$.*
3. *If a system switched from g_n to some other controller at time t_{off} and if t_{on} is the time when the system next switches again to g_n , then $V(t_{\text{off}}) \geq V(t_{\text{on}})$.*

Then the submanifold E_n is globally attractive.

Remark: The first condition is the same as in Proposition 5, while conditions (2) and (3) together replace condition (2) there. Condition (2) says that for any manifold M_i on which a trajectory stays for an infinite amount of time, we can find a switch at an arbitrary large time to a manifold that lies lower in the hierarchy implied by \preceq and that after such switch the system evolves on the manifolds that are below M_i for at least Δ .

Proof. We will show that conditions (2) and (3) imply condition (2) of Proposition 5. Let $x(t)$ be a trajectory of Σ and let M_i be a manifold on which $x(t)$ evolves for an infinite amount of time. Since we have a finite number of manifolds, there will be at least one such i . The condition (2) guarantees that there will be an infinite number of instances when the system evolves for at least Δ on manifolds that are below M_i in the hierarchy defined by \preceq . But this implies that $x(t)$ will evolve on these manifolds for an infinite amount of time and since there are only finitely many manifolds below M_i , there must exist a manifold M_j with $j \prec i$ on which $x(t)$ evolves for an infinite amount of time. By proceeding recursively and because n is the smallest element for \preceq , we conclude that the system must evolve on M_n for an infinite amount of time and in instances that last for at least Δ . Condition (3) guarantees that each time the system switches to g_n , the value of the Lyapunov function is smaller than when the system last switched off M_n . The existence of the sequence $\{t_i\}$ in condition (2) of Proposition 5 is therefore guaranteed.

Using Proposition 6 we can design a stable switching scheme by choosing a partial order, developing controllers on each M_i that guarantee a switch to a lower level with respect to this partial order, and enforcing decreasing of V at switches to M_n . However, developing controllers that guarantee a switch to a lower level is still not an easy task. One possible strategy is to make each controller stabilize a certain manifold within a region from which the system switches to manifolds lower in the hierarchy. This special case is important enough that we state a separate proposition.

Proposition 7. Assume a partial order \preceq on A that has the smallest element which is equal to n . Let each controller g_i asymptotically stabilize a manifold E_i and assume we can find a Lyapunov function V_i for g_i . Let the switching scheme S satisfy the following conditions:

1. For each i , there exists $L_i > 0$ such that $S(x, i) \prec i$ for every $x \in B(E_i, L_i) \cap M_i$ (for $i = n$ we require $S(x, n) = n$).
2. There exists $\Delta > 0$, such that if a system switches from g_i to some g_j , $j \prec i$ at time T , then $S(x(t), \eta(t)) \prec i$ for each $t \in [T, T + \Delta]$.
3. If the system switches from g_i to some g_j , $i \prec j$, at time t_{off} and after that switches again to g_i at time t_{on} and if $S(x(t), \eta(t)) \not\prec i$ for all $t \in [t_{\text{off}}, t_{\text{on}}]$, then $V_i(t_{\text{off}}) \geq V_i(t_{\text{on}})$.

Then the submanifold E_n is globally attractive.

Remark: For $i = n$ conditions (1) and (3) above clearly become the same as conditions (1) and (3) in Proposition 6. Note that the Proposition suggests that we can examine the stability of the system by simply examining relations between neighbors defined by the switching scheme. This has important implications for the synthesis problem and can be explored to obtain modularity of the design process.

Proof. We will show that the above conditions imply conditions of the Proposition 6. Assume that a trajectory $x(t)$ evolves on a manifold M_i for an infinite amount of time, but after some time T it never switches to any manifold M_j such that $j \prec i$. Let $I_i = \{t > T \mid S(x(t), \eta(t)) = i\}$, the union of the intervals beyond T during which the system evolves on M_i . By condition (3), V_i will be monotonically decreasing on I_i and by condition (2), we can find an infinite sequence of (disjoint) intervals of length Δ that lie in I_i . By the same reasoning that we used in the proof of Proposition 5 to show convergence to E_n we can show that $x(t)$ converges to E_i . By condition (1) this implies that the system will switch to some M_j , $j \prec i$, which is a contradiction. This and condition (2) above therefore imply condition (2) of Proposition 6.

The last proposition is a convenient tool for designing stable switching control schemes. The algorithm for controller design can be roughly described as:

- Choose a partial order on A (decide on the hierarchy among M_i 's).
- Design a controller on each M_i that stabilizes a manifold E_i .
- Choose a neighborhood U_i of E_i and define a switching scheme so that for $x \in U_i$, $S(x, i) \prec i$.

Clearly, this basic algorithm has to be refined to guarantee that the conditions (2) and (3) above are satisfied.

There is an important case in which condition (2) can be satisfied fairly easily. Suppose we want to switch from M_i to M_j , $j \prec i$. If $f_j(x, g_j(x))$ in Eq. (2) is bounded for all $x \in U \subset M_j$, where U is a neighborhood that contains the region to which the system switches, then all we need to do is make the system switch in such a way that after the switch to M_j we are some (fixed) finite distance away from any point x in U for which $j \prec S(x, j)$. Because of the bounded rate of change of the state, this implies that the switch will occur after some finite time interval.

It is difficult to directly design controllers that would satisfy condition (3). An alternative is to combine several controllers, each of which partly satisfies the condition, into a single controller. Suppose we would like to allow switches from M_i to M_j , $j \prec i$. To satisfy condition (3), we need to have a controller g_i that is able to decrease the Lyapunov function V_j . Controller g_i stabilizes E_i , and we also know that the controller g_j decreases the Lyapunov function V_j . If E_i is the equilibrium manifold for the system controlled by g_i , we can construct a new controller, \hat{g}_i that behaves as g_i away from E_i and as g_j close to E_i . A possible expression for \hat{g}_i would be:

$$\hat{g}_i(x) = (1 - c_1 e^{-c_2 d(x, E_1)})g_i(x) + c_1 e^{-c_2 d(x, E_1)}g_j(x) \quad (7)$$

where c_1 and c_2 are appropriate constants.

Propositions 5-7 provide sufficient conditions for E_n to be attractive, not to be stable. To prove the stability we have to show that trajectories starting outside M_n "nicely" converge to M_n . One possible way of stating this is:

Corollary 8. *The manifold E_n will be stable if in addition to the conditions of Proposition 5:*

- (3) *For any $R > 0$ and every i , there exists $r > 0$ such that if $x(t_0) \in (M_i \setminus M_n) \cap B(E_n, r)$ then under the control of g_i , $x(t) \in B(E_n, R)$ for every $t > t_0$.*

Proof. The Lyapunov function V guarantees that for any $R_2 > 0$, there exists $r_2 > 0$ such that $x(t_0) \in M_n \cap B(E_2, r_2)$ implies $x(t) \in B(E_n, R_2)$ as long as $x(t)$ stays in M_n . Take $R_2 = \min\{R, L\}$ and find the corresponding r_2 . Take $R_1 = \min\{R, r_2\}$. By assumption, there exists r_1 such that $x(t)$ stays in $B(E_n, R_1)$ for any trajectory starting in $B(E_n, r_1) \setminus M_n$ and evolving in M_i . By condition (1) of Proposition 5 and by the choice of R_1 , $x(t)$ will intersect M_n inside $B(E_n, r_2) \cap M_n$. But a trajectory on M_n that comes inside $B(E_n, r_2) \cap M_n$ will stay inside $B(E_n, L) \cap M_n$ and thus remain under the control of g_n (and stay inside $B(E_n, R)$) for all later times.

We note that this proof is similar to the proof of Theorem 4 in [17].

Remark If we assume the scenario of Proposition 7 and for every i , $E_i \subseteq E_n$, the condition of the Corollary will be trivially true.

4 Example

The above results provide a framework for designing hybrid control schemes. In this section we apply the methodology to stabilization of the shimmying wheel (Fig. 1). Dynamic equations of the system are of the form:

$$H \begin{bmatrix} \ddot{y} \\ \ddot{\theta} \\ \ddot{\phi} \end{bmatrix} + \begin{bmatrix} ky + \frac{l}{2}(m_1 + 2m_2)\dot{\theta}^2 \sin \theta \\ 0 \\ 0 \end{bmatrix} = A^T F + \begin{bmatrix} 0 \\ u \\ 0 \end{bmatrix} \quad (8)$$

where H is the inertia matrix, $F = \{F_x, F_y\}^T$ is the reaction force of the ground on the wheel, and A is the matrix that relates the relative velocity v_r between the wheel and the ground at the contact point to the rate of change of the generalized coordinates. The system has 6 states: 3 generalized coordinates and 3 generalized velocities.

When the wheel is sliding, we have the following expression for the reaction force $F = F_s$:

$$F_s = -\mu_d \frac{v_r}{\|v_r\|} (m_1 + \frac{m_2}{2})g \quad (9)$$

where μ_d is the coefficient of (dynamic) friction and g is the gravity constant. When the wheel is rolling, we have an additional constraint:

$$v_r = 0 \quad (10)$$

In this case, the force $F = F_c$ is the constraint force that prevents slippage of the wheel and it can be eliminated from Eq. (8) using Eq. (10) [21, 22]. Equation (10) represents two constraint equations, so the dimension of the system in pure rolling drops to 4. The analysis of the system can be simplified by observing that ϕ does not occur in the dynamic equations. It is therefore a cyclic variable and we can limit our study to the dynamics of y and θ . In the formalism of Section 2, the reduced system thus evolves on manifolds M_1 and M_2 of dimension 4 and 3, respectively, where $M_1 = \mathbb{R}^4$ and M_2 is defined by Eq. (10) [21, 22].

The goal of the control is to stabilize the wheel to the state $y = 0, \theta = 0$. To this end, we introduce an additional region, M_3 , but we put $M_3 = M_2$. In other words, we use two different controllers in the rolling regime. Note that nothing in the developed theory prohibits the submanifolds to be equal. Stabilization is therefore achieved with three controllers: a controller g_1 for the system in sliding regime (defined on M_1) and controllers g_2 and g_3 for the system in the rolling mode (defined on M_2). The idea is to steer the system with the controllers g_1 and g_2 to a state $\theta = 0, y \neq 0$, from which we can stabilize the system to a desired point with the controller g_3 . Note that the wheel might start sliding again once under the control of g_3 .

To design a controller for the system evolving on M_1 , we linearize the dynamic response for θ . It can be shown that with this controller the dynamics for y and ϕ are also (asymptotically) stable. The controller stabilizes the line segment:

$$E_1 = (y, 0, 0, 0) \quad |y| \leq \frac{\mu_d(m_1 + 2m_2)g}{2k}$$

The controller g_2 (only defined on M_2 , when the wheel is rolling) can be designed similarly to g_1 after the constraint force is eliminated from dynamic equations using Eq. (10). The attractive manifold for this controller is a line:

$$E_2 = (y, \theta, \dot{y}, \dot{\theta}) = (y, 0, 0, 0)$$

The controller g_3 can be derived by observing that instead of the dynamics for θ , we can linearize the dynamics for y . Further analysis shows that with this controller, the dynamics for θ and ϕ are stable, so the system converges to the desired point, $E_3 = (0, 0, 0, 0)$. It is also not difficult to construct the Lyapunov functions V_2 and V_3 for the controllers g_2 and g_3 .

Next, we have to define a partial order and design the switching schemes. We first observe that there is a natural partial order already defined on $\mathcal{M} = \{M_1, M_2, M_3\}$ and it is given by inclusion: $M_1 \supset M_2 \supseteq M_3$. The partial order in this case thus becomes a total order and the application of Proposition 7 is therefore particularly straight forward.

The switching scheme \mathcal{S}_1 is quite simple:

$$\mathcal{S}_1(x, \eta) = \begin{cases} 2 & x \in M_2 \wedge \|F_c\| \leq \frac{\mu_d}{2}(m_1 + 2m_2)g \\ 1 & \text{otherwise} \end{cases}$$

The controller g_2 has a singularity at $\theta = \pm \frac{\pi}{2}$, but on these two hyperplanes the constraint force is unbounded and they do not intersect (the closure of) M_2 . The switching scheme \mathcal{S}_2 is defined in the following way:

$$\mathcal{S}_2(x, \eta) = \begin{cases} 3 & \eta = 2 \wedge x \in B(E_3, R_{in}) \wedge V_3(x) \leq V_3^{3 \rightarrow 2} \\ & \wedge \|F_c\| \leq \frac{\mu_d}{4}(m_1 + 2m_2)g \\ 3 & \eta = 3 \wedge x \in B(E_3, R_{out}) \\ 2 & \text{otherwise} \end{cases}$$

where $R_{in} < R_{out} < \frac{\pi}{2}$ (this guarantees that $B(E_3, R_{out})$ does not intersect the hyperplanes $\theta = \pm \frac{\pi}{2}$), and $V_3^{3 \rightarrow 2}$ is the value of V_3 when the system last switched from the controller g_3 to the controller g_2 . Again, we avoid the hyperplanes $\theta = \pm \frac{\pi}{2}$ because g_3 becomes singular there. Observe that the switching scheme explicitly encodes condition (3) of Proposition 7.

The next step would be to check that the conditions of the Proposition 7 are satisfied. Since we have a total order on \mathcal{M} , it suffices to show that g_1 and g_2 stabilize E_2 , and that g_2 and g_3 stabilize E_3 . In the interest of keeping the presentation short the proofs will be omitted, but we refer the interested reader to [26] for details. We only mention that in order to guarantee that the controller g_2 can arbitrarily decrease the Lyapunov function V_3 so that the system can switch to g_3 , we use the technique described in Eq. (7).

4.1 Simulation results

A typical simulation run of the system controlled with the derived controllers is shown in Fig. 5. The system starts in the sliding regime with the controller g_1 active. At 0.9s the wheel stops sliding and the controller g_2 takes over. At 1.14s the system switches again, this time to the controller g_3 that stabilizes the system to the desired state. The switches between different controllers cause discontinuities of the input, as Fig. 5(b). shows. It can be seen in Fig. 5(a) that while the controllers g_1 and g_2 are active, θ is the controlled variable and it decreases to 0. When the controller g_3 becomes active, the controlled variable becomes y (so it decreases to 0) and $|\theta|$ initially increases. After y becomes small, $|\theta|$ also decreases to 0.

The next figure illustrates that the modified controller \hat{g}_2 decreases the Lyapunov function V_3 . Variables y and θ are shown in Fig. 6(a), while the Lyapunov functions V_2 and V_3 are shown in Fig. 6(b). The system starts in the rolling regime with the controller g_3 active, however during the first 0.1s it switches first to the controller g_2 and then to the sliding regime and the controller g_1 (these switches are not shown). At the switch from g_3 to g_2 the value of the Lyapunov function V_3 is 263.4. To show that the controller can arbitrary decrease

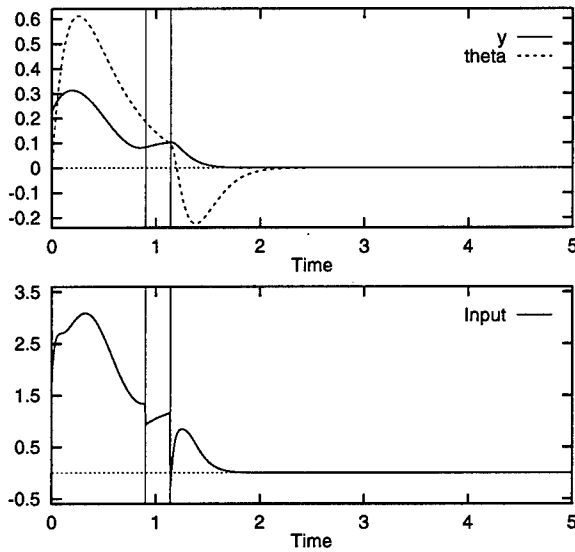


Fig. 5. A typical simulation run.

V_3 , we modified the switching scheme S_2 so that the value of the Lyapunov function V_3 at the switch from g_2 to g_3 has to be half the value of the function at the switch from g_3 to g_2 . In our case, the function V_3 therefore has to decrease to 131.7 in order to switch to the controller g_3 . At the time 0.38s, the system switches from sliding to rolling and to the controller g_2 . The controller decreases the Lyapunov function until it reaches the desired value at the time 1.30s when the system switches to the controller g_3 and the system is stabilized. Figure 6(a) also shows that the controller g_2 does not drive θ to 0 but to some offset value that guarantees the decreasing of V_3 .

5 Conclusion

We investigated the problem of stabilizing a system with changing dynamics with a sequence of controllers. We studied the case when the system evolves on a sequence of embedded manifolds and derived sufficient conditions under which the switching scheme employing different controllers can be guaranteed to stabilize the system to the desired manifold. These sufficient conditions give direct guidance for the design of appropriate controllers. The results were applied to the stabilization of the shimmying wheel. We were able to design a switching scheme that provably stabilizes this system.

The described work can be extended in several directions. We plan to consider more general stabilization problems such as control of a walking robot. In this case, the system has to be stabilized to a periodic orbit that traverses different

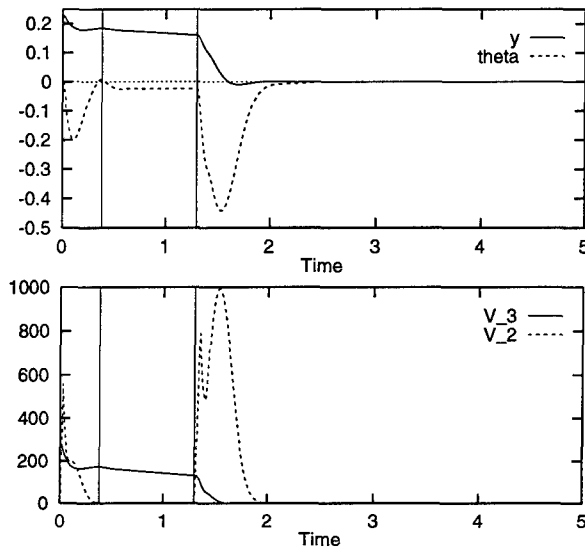


Fig. 6. A modified controller guarantees decreasing of V_3 .

regions rather than an equilibrium manifold within a single region. An important question is also how to design the individual controllers. For mechanical systems, the energy-momentum method offers some interesting possibilities.

References

1. R. Alur, C. Courcoubetis, T. Henzinger, and P. H. Ho, "Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems," in *LNCS 736*, pp. 209–229, Springer-Verlag, 1993.
2. X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "An approach to the description and analysis of hybrid systems," in *LNCS 736*, pp. 149–178, Springer-Verlag, 1993.
3. R. W. Brockett, "Hybrid models for motion control systems," in *Essays in Control: Perspectives in the Theory and its Applications*, pp. 29–53, Boston: Birkhäuser, 1993.
4. A. Nerode and W. Kohn, "Models for hybrid systems: Automata, topologies, stability," in *LNCS 736*, pp. 317–356, Springer-Verlag, 1993.
5. A. Back, J. Guckenheimer, and M. Myers, "A dynamical simulation facility for hybrid systems," in *LNCS 736*, pp. 255–267, Springer-Verlag, 1993.
6. M. S. Branicky, V. S. Borkar, and S. K. Mitter, "A unified framework for hybrid control," in *Proceedings of the 33rd IEEE Conference on Decision and Control*, (Lake Buena Vista, FL), pp. 4228–4234, 1994.
7. J. Lygeros, D. N. Godbole, and S. S. Sastry, "A game theoretic approach to hybrid system design," in *LNCS 1066*, pp. 1–12, Springer-Verlag, 1996.
8. A. Puri, *Theory of hybrid systems and discrete event systems*. PhD thesis, U. C. Berkeley, 1995.

9. A. Deshpande and P. Varaiya, "Viable control of hybrid systems," in *LNCS 999*, pp. 128–147, Springer-Verlag, 1995.
10. W. Kohn, A. Nerode, J. B. Remmel, and X. Ge, "Multiple agent hybrid control: carrier manifolds and chattering approximations to optimal control," in *Proceedings of the 33rd IEEE Conference on Decision and Control*, (Lake Buena Vista, FL), pp. 4221–4227, 1994.
11. M. S. Branicky and S. K. Mitter, "Algorithms for optimal hybrid control," in *Proceedings of the 34th IEEE Conference on Decision and Control*, (New Orleans, LA), pp. 2661–2666, 1995.
12. M. Žefran, J. Desai, and V. Kumar, "Continuous motion plans for robotic systems with changing dynamic behavior," in *Robotic motion and manipulation*, pp. 113–128, Wellesley, MA: A K Peters, 1997.
13. B. Goodwine and J. W. Burdick, "A general method for motion planning for quasi-static legged robotic locomotion." Preprint, 1997.
14. T. Henzinger, Z. Manna, and A. Pnueli, "Temporal proof methodologies for timed transition systems," *Inf. and Comp.*, vol. 112, no. 2, pp. 273–337, 1994.
15. T. Henzinger, P. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata," in *27th Ann. ACM Symp. on the Theory of Computing*, 1995.
16. N. Lynch, "Modelling and verification of automated transit systems, using timed automata, invariants and simulations," in *LNCS 1066*, pp. 449–463, Springer-Verlag, 1996.
17. M. S. Branicky, "Stability of switched and hybrid systems," in *Proceedings of the 33rd IEEE Conference on Decision and Control*, (Lake Buena Vista, FL), pp. 3498–3503, 1994.
18. J. Malmborg, B. M. Bernhardsson, and K. J. Åström, "A stabilizing switching scheme for multi-controller systems," in *13th IFAC World Congress*, (San Francisco, CA), 1996.
19. P. Peleties and R. DeCarlo, "Asymptotic stability of m-switched systems using Lyapunov-like functions," in *American Control Conf.*, (Boston), pp. 1679–1684, 1991.
20. R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, "Sequential composition of dynamically dexterous robot behaviors." Preprint, 1996.
21. G. Stépán, "Chaotic motion of wheels," *Vehicle System Dynamics*, vol. 20, pp. 341–351, 1991.
22. B. Goodwine and G. Stépán, "Controlling unstable rolling phenomena." To appear in the *Journal of Vibration and Control*, 1997.
23. W. Hahn, *Stability of motion*. Springer-Verlag, 1967.
24. N. Rouche, P. Habets, and M. LaLoy, *Stability theory by Liapunov's direct method*. New York: Springer-Verlag, 1977.
25. A. Vannelli and M. Vidyasagar, "Theory of partial stability theorems, converse theorems, and maximal Lyapunov functions," in *Proc. Annu. Southeast Symp. Syst. Theory*, (Piscataway, NJ), pp. 16–20, 1980.
26. M. Žefran and J. W. Burdick, "Switching control on embedded manifolds," tech. rep., Caltech, 1997.

List of Authors

E. Asarin	1	O. Maler	96
A. Balluchi	13	Z. Manna	305
O. Beldiman	64	A.S. Matveev	319
A. Beydoun	33	B.M. Miller	334
S. Bornot	49	I. Mitchell	159
J.W. Burdick	400	R. Naumann	221
L. Bushnell	64	T.W. Neller	346
P.E. Caines	237	G.J. Pappas	205, 289
B. Carlson	80	C. Pinello	13
T. Dang	96	R. Rasche	221
M. Di Benedetto	13	C. Rossi	13
A. Fehnker	110	C. Rust	221
E.D. Ferreira	126	V. Rusu	190
V. Friesen	143	A. Sangiovanni-Vincentelli	13
M.R. Greenstreet	159	S. Sastry	205, 289, 360
V. Gupta	80	A.V. Savkin	319
K.X. He	175	J.H. van Schuppen	374
T.A. Henzinger	190	J. Sifakis	49
J. Hou	384	H.B. Sipma	305
B.H. Krogh	126	S. Sivashankar	33
G. Lafferriere	205	J. Sun	33
G. Lehrenfeld	221	J. Tacken	221
E.S. Lemch	237	C. Tomlin	360
M.D. Lemmon	175	G. Walsh	64
X. Li	384	L.Y. Wang	33
C. Livadas	253	M. Žefran	400
J. Lygeros	273, 289, 360	J. Zhao	384
N.A. Lynch	253, 273	G. Zheng	384
		T. Zheng	384

Springer and the environment

At Springer we firmly believe that an international science publisher has a special obligation to the environment, and our corporate policies consistently reflect this conviction.

We also expect our business partners – paper mills, printers, packaging manufacturers, etc. – to commit themselves to using materials and production processes that do not harm the environment. The paper in this book is made from low- or no-chlorine pulp and is acid free, in conformance with international standards for paper permanency.



Springer

Lecture Notes in Computer Science

For information about Vols. 1–1308

please contact your bookseller or Springer-Verlag

Vol. 1309: R. Steinmetz, L.C. Wolf (Eds.), *Interactive Distributed Multimedia Systems and Telecommunication Services. Proceedings, 1997. XIII, 466 pages. 1997.*

Vol. 1310: A. Del Bimbo (Ed.), *Image Analysis and Processing. Proceedings, 1997. Volume I. XXII, 722 pages. 1997.*

Vol. 1311: A. Del Bimbo (Ed.), *Image Analysis and Processing. Proceedings, 1997. Volume II. XXII, 794 pages. 1997.*

Vol. 1312: A. Geppert, M. Berndtsson (Eds.), *Rules in Database Systems. Proceedings, 1997. VII, 214 pages. 1997.*

Vol. 1313: J. Fitzgerald, C.B. Jones, P. Lucas (Eds.), *FME '97: Industrial Applications and Strengthened Foundations of Formal Methods. Proceedings, 1997. XIII, 685 pages. 1997.*

Vol. 1314: S. Muggleton (Ed.), *Inductive Logic Programming. Proceedings, 1996. VIII, 397 pages. 1997. (Subseries LNAI).*

Vol. 1315: G. Sommer, J.J. Koenderink (Eds.), *Algebraic Frames for the Perception-Action Cycle. Proceedings, 1997. VIII, 395 pages. 1997.*

Vol. 1316: M. Li, A. Maruoka (Eds.), *Algorithmic Learning Theory. Proceedings, 1997. XI, 461 pages. 1997. (Subseries LNAI).*

Vol. 1317: M. Leman (Ed.), *Music, Gestalt, and Computing. IX, 524 pages. 1997. (Subseries LNAI).*

Vol. 1318: R. Hirschfeld (Ed.), *Financial Cryptography. Proceedings, 1997. XI, 409 pages. 1997.*

Vol. 1319: E. Plaza, R. Benjamins (Eds.), *Knowledge Acquisition, Modeling and Management. Proceedings, 1997. XI, 389 pages. 1997. (Subseries LNAI).*

Vol. 1320: M. Mavronicolas, P. Tsigas (Eds.), *Distributed Algorithms. Proceedings, 1997. X, 333 pages. 1997.*

Vol. 1321: M. Lenzerini (Ed.), *AI*IA 97: Advances in Artificial Intelligence. Proceedings, 1997. XII, 459 pages. 1997. (Subseries LNAI).*

Vol. 1322: H. Hußmann, *Formal Foundations for Software Engineering Methods. X, 286 pages. 1997.*

Vol. 1323: E. Costa, A. Cardoso (Eds.), *Progress in Artificial Intelligence. Proceedings, 1997. XIV, 393 pages. 1997. (Subseries LNAI).*

Vol. 1324: C. Peters, C. Thanos (Eds.), *Research and Advanced Technology for Digital Libraries. Proceedings, 1997. X, 423 pages. 1997.*

Vol. 1325: Z.W. Raś, A. Skowron (Eds.), *Foundations of Intelligent Systems. Proceedings, 1997. XI, 630 pages. 1997. (Subseries LNAI).*

Vol. 1326: C. Nicholas, J. Mayfield (Eds.), *Intelligent Hypertext. XIV, 182 pages. 1997.*

Vol. 1327: W. Gerstner, A. Germond, M. Hasler, J.-D. Nicoud (Eds.), *Artificial Neural Networks – ICANN '97. Proceedings, 1997. XIX, 1274 pages. 1997.*

Vol. 1328: C. Retoré (Ed.), *Logical Aspects of Computational Linguistics. Proceedings, 1996. VIII, 435 pages. 1997. (Subseries LNAI).*

Vol. 1329: S.C. Hirtle, A.U. Frank (Eds.), *Spatial Information Theory. Proceedings, 1997. XIV, 511 pages. 1997.*

Vol. 1330: G. Smolka (Ed.), *Principles and Practice of Constraint Programming – CP 97. Proceedings, 1997. XII, 563 pages. 1997.*

Vol. 1331: D. W. Embley, R. C. Goldstein (Eds.), *Conceptual Modeling – ER '97. Proceedings, 1997. XV, 479 pages. 1997.*

Vol. 1332: M. Bubak, J. Dongarra, J. Waśniewski (Eds.), *Recent Advances in Parallel Virtual Machine and Message Passing Interface. Proceedings, 1997. XV, 518 pages. 1997.*

Vol. 1333: F. Pichler, R. Moreno-Díaz (Eds.), *Computer Aided Systems Theory – EUROCAST'97. Proceedings, 1997. XII, 626 pages. 1997.*

Vol. 1334: Y. Han, T. Okamoto, S. Qing (Eds.), *Information and Communications Security. Proceedings, 1997. X, 484 pages. 1997.*

Vol. 1335: R.H. Möhring (Ed.), *Graph-Theoretic Concepts in Computer Science. Proceedings, 1997. X, 376 pages. 1997.*

Vol. 1336: C. Polychronopoulos, K. Joe, K. Araki, M. Amamiya (Eds.), *High Performance Computing. Proceedings, 1997. XII, 416 pages. 1997.*

Vol. 1337: C. Freksa, M. Jantzen, R. Valk (Eds.), *Foundations of Computer Science. XII, 515 pages. 1997.*

Vol. 1338: F. Plášil, K.G. Jeffery (Eds.), *SOFSEM'97: Theory and Practice of Informatics. Proceedings, 1997. XIV, 571 pages. 1997.*

Vol. 1339: N.A. Murshed, F. Bortolozzi (Eds.), *Advances in Document Image Analysis. Proceedings, 1997. IX, 345 pages. 1997.*

Vol. 1340: M. van Kreveld, J. Nievergelt, T. Roos, P. Widmayer (Eds.), *Algorithmic Foundations of Geographic Information Systems. XIV, 287 pages. 1997.*

Vol. 1341: F. Bry, R. Ramakrishnan, K. Ramamohanarao (Eds.), *Deductive and Object-Oriented Databases. Proceedings, 1997. XIV, 430 pages. 1997.*

Vol. 1342: A. Sattar (Ed.), *Advanced Topics in Artificial Intelligence. Proceedings, 1997. XVII, 516 pages. 1997. (Subseries LNAI).*

Vol. 1343: Y. Ishikawa, R.R. Oldehoeft, J.V.W. Reynnders, M. Tholburn (Eds.), *Scientific Computing in Object-Oriented Parallel Environments. Proceedings, 1997. XI, 295 pages. 1997.*

- Vol. 1344: C. Ausnit-Hood, K.A. Johnson, R.G. Pettit, IV, S.B. Opdahl (Eds.), *Ada 95 – Quality and Style*. XV, 292 pages. 1997.
- Vol. 1345: R.K. Shyamasundar, K. Ueda (Eds.), *Advances in Computing Science - ASIAN'97. Proceedings, 1997*. XIII, 387 pages. 1997.
- Vol. 1346: S. Ramesh, G. Sivakumar (Eds.), *Foundations of Software Technology and Theoretical Computer Science. Proceedings, 1997*. XI, 343 pages. 1997.
- Vol. 1347: E. Ahronovitz, C. Fiorio (Eds.), *Discrete Geometry for Computer Imagery. Proceedings, 1997*. X, 255 pages. 1997.
- Vol. 1348: S. Steel, R. Alami (Eds.), *Recent Advances in AI Planning. Proceedings, 1997*. IX, 454 pages. 1997. (Subseries LNAI).
- Vol. 1349: M. Johnson (Ed.), *Algebraic Methodology and Software Technology. Proceedings, 1997*. X, 594 pages. 1997.
- Vol. 1350: H.W. Leong, H. Imai, S. Jain (Eds.), *Algorithms and Computation. Proceedings, 1997*. XV, 426 pages. 1997.
- Vol. 1351: R. Chin, T.-C. Pong (Eds.), *Computer Vision – ACCV'98. Proceedings Vol. I, 1998*. XXIV, 761 pages. 1997.
- Vol. 1352: R. Chin, T.-C. Pong (Eds.), *Computer Vision – ACCV'98. Proceedings Vol. II, 1998*. XXIV, 757 pages. 1997.
- Vol. 1353: G. BiBattista (Ed.), *Graph Drawing. Proceedings, 1997*. XII, 448 pages. 1997.
- Vol. 1354: O. Burkart, *Automatic Verification of Sequential Infinite-State Processes*. X, 163 pages. 1997.
- Vol. 1355: M. Darnell (Ed.), *Cryptography and Coding. Proceedings, 1997*. IX, 335 pages. 1997.
- Vol. 1356: A. Danthine, Ch. Diot (Eds.), *From Multimedia Services to Network Services. Proceedings, 1997*. XII, 180 pages. 1997.
- Vol. 1357: J. Bosch, S. Mitchell (Eds.), *Object-Oriented Technology. Proceedings, 1997*. XIV, 555 pages. 1998.
- Vol. 1358: B. Thalheim, L. Libkin (Eds.), *Semantics in Databases*. XI, 265 pages. 1998.
- Vol. 1360: D. Wang (Ed.), *Automated Deduction in Geometry. Proceedings, 1996*. VII, 235 pages. 1998. (Subseries LNAI).
- Vol. 1361: B. Christianson, B. Crispo, M. Lomas, M. Roe (Eds.), *Security Protocols. Proceedings, 1997*. VIII, 217 pages. 1998.
- Vol. 1362: D.K. Panda, C.B. Stunkel (Eds.), *Network-Based Parallel Computing. Proceedings, 1998*. X, 247 pages. 1998.
- Vol. 1363: J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer, D. Snyers (Eds.), *Artificial Evolution*. XI, 349 pages. 1998.
- Vol. 1364: W. Conen, G. Neumann (Eds.), *Coordination Technology for Collaborative Applications*. VIII, 282 pages. 1998.
- Vol. 1365: M.P. Singh, A. Rao, M.J. Wooldridge (Eds.), *Intelligent Agents IV. Proceedings, 1997*. XII, 351 pages. 1998. (Subseries LNAI).
- Vol. 1367: E.W. Mayr, H.J. Prömel, A. Steger (Eds.), *Lectures on Proof Verification and Approximation Algorithms*. XII, 344 pages. 1998.
- Vol. 1368: Y. Masunaga, T. Katayama, M. Tsukamoto (Eds.), *Worldwide Computing and Its Applications – WWC'A'98. Proceedings, 1998*. XIV, 473 pages. 1998.
- Vol. 1370: N.A. Streitz, S. Konomi, H.-J. Burkhardt (Eds.), *Cooperative Buildings. Proceedings, 1998*. XI, 267 pages. 1998.
- Vol. 1372: S. Vaudenay (Ed.), *Fast Software Encryption. Proceedings, 1998*. VIII, 297 pages. 1998.
- Vol. 1373: M. Morvan, C. Meinel, D. Krob (Eds.), *STACS 98. Proceedings, 1998*. XV, 630 pages. 1998.
- Vol. 1374: H. Bunt, R.-J. Beun, T. Borghuis (Eds.), *Multimodal Human-Computer Communication*. VIII, 345 pages. 1998. (Subseries LNAI).
- Vol. 1375: R. D. Hersch, J. André, H. Brown (Eds.), *Electronic Publishing, Artistic Imaging, and Digital Typography. Proceedings, 1998*. XIII, 575 pages. 1998.
- Vol. 1376: F. Parisi Presicce (Ed.), *Recent Trends in Algebraic Development Techniques. Proceedings, 1997*. VIII, 435 pages. 1998.
- Vol. 1377: H.-J. Schek, F. Salto, I. Ramos, G. Alonso (Eds.), *Advances in Database Technology – EDBT'98. Proceedings, 1998*. XII, 515 pages. 1998.
- Vol. 1378: M. Nivat (Ed.), *Foundations of Software Science and Computation Structures. Proceedings, 1998*. X, 289 pages. 1998.
- Vol. 1379: T. Nipkow (Ed.), *Rewriting Techniques and Applications. Proceedings, 1998*. X, 343 pages. 1998.
- Vol. 1380: C.L. Lucchesi, A.V. Moura (Eds.), *LATIN'98: Theoretical Informatics. Proceedings, 1998*. XI, 391 pages. 1998.
- Vol. 1381: C. Hankin (Ed.), *Programming Languages and Systems. Proceedings, 1998*. X, 283 pages. 1998.
- Vol. 1382: E. Astesiano (Ed.), *Fundamental Approaches to Software Engineering. Proceedings, 1998*. XII, 331 pages. 1998.
- Vol. 1383: K. Koskimies (Ed.), *Compiler Construction. Proceedings, 1998*. X, 309 pages. 1998.
- Vol. 1384: B. Steffen (Ed.), *Tools and Algorithms for the Construction and Analysis of Systems. Proceedings, 1998*. XIII, 457 pages. 1998.
- Vol. 1385: T. Margaria, B. Steffen, R. Rückert, J. Posegga (Eds.), *Services and Visualization. Proceedings, 1997/1998*. XII, 323 pages. 1998.
- Vol. 1386: T.A. Henzinger, S. Sastry (Eds.), *Hybrid Systems: Computation and Control. Proceedings, 1998*. VIII, 417 pages. 1998.
- Vol. 1387: C. Lee Giles, M. Gori (Eds.), *Adaptive Processing of Sequences and Data Structures. Proceedings, 1997*. XII, 434 pages. 1998. (Subseries LNAI).
- Vol. 1388: J. Rolim (Ed.), *Parallel and Distributed Processing. Proceedings, 1998*. XVII, 1168 pages. 1998.
- Vol. 1389: K. Tombre, A.K. Chhabra (Eds.), *Graphics Recognition. Proceedings, 1997*. XII, 421 pages. 1998.
- Vol. 1391: W. Banzhaf, R. Poli, M. Schoenauer, T.C. Fogarty (Eds.), *Genetic Programming. Proceedings, 1998*. X, 232 pages. 1998.

Lecture Notes in Computer Science

This series reports new developments in computer science research and teaching, quickly, informally, and at a high level. The timeliness of a manuscript is more important than its form, which may be unfinished or tentative. The type of material considered for publication includes

- drafts of original papers or monographs,
- technical reports of high quality and broad interest,
- advanced-level lectures,
- reports of meetings, provided they are of exceptional interest and focused on a single topic.

Publication of Lecture Notes is intended as a service to the computer science community in that the publisher Springer-Verlag offers global distribution of documents which would otherwise have a restricted readership. Once published and copyrighted they can be cited in the scientific literature.

Manuscripts

Lecture Notes are printed by photo-offset from the master copy delivered in camera-ready form. Manuscripts should consist of no fewer than 100 and preferably no more than 500 pages of text. Authors of monographs and editors of proceedings volumes receive 50 free copies of their book. Manuscripts should be printed with a laser or other high-resolution printer onto white paper of reasonable quality. To ensure that the final photo-reduced pages are easily readable, please use one of the following formats:

Front size (points)	Printing area		Final size %
	(cm)	(inches)	
10	12.2 x 19.3	4.8 x 7.6	100
12	15.3 x 24.2	6.0 x 9.5	80

On request the publisher will supply a leaflet with more detailed technical instructions or a T_EX macro package for the preparation of manuscripts.

Manuscripts should be sent to one of the series editors or directly to:

Springer-Verlag, Computer Science Editorial III, Tiergartenstr.17,
D-69121 Heidelberg, Germany

ISSN 0302-9743

<http://www.springer.de>